SENIOR DESIGN PROJECT REPORT

Pet Feeder with Pet Identification

Submitted to

Dr. Robert Weissbach
Engineering Technology Department
by

Demetric Taylor, Aditan Kodjo, Khalid Bamusa

December 14, 2019

Pet Feeder with Pet Identification

EXECUTIVE SUMMARY

The group was given the task of designing and building an automated pet feeder that would use a form of pet identification. The pet feeder would be used by a cat, so it would need to cater directly towards cats. Dr. Weissbach, who was our sponsor as well as the advisor for this project wanted us to design this system for his own cat, Lucas. Meeting with him about his needs for the feeder, we were to have a robust system This would need to be something that's easy to operate. It was to also be light in weight when there isn't food inside the reservoir. After getting this information from our customer, we began to dive into the project. As we started to plan out the feeder, we decided on some components that would come in handy to have a working machine. Using such components as an ultrasonic sensor, voice recognition module, and servo motor all seemed to be the right devices to incorporate in the machine. When the feeder was finished being constructed and we moved onto testing, we have some interesting results that I'll get into later in the reading. Some recommendations for system improvements would deal with our food dispenser and our servo motor.

Pet Feeder with Pet Identification

TABLE OF CONTENTS

Pet Feeder with Pet Identification

REVISION HISTORY

Version	Date	Revised by	Description
1.0	26 November 2019	Demetric Taylor, Aditan Kodjo, Khalid Bamusa	Initial version
2.0	14 December 2019	Demetric Taylor, Aditan Kodjo, Khalid Bamusa	Revised for final submission

Pet Feeder with Pet Identification

INTRODUCTION

The group members were given the task of designing and building an automatic cat feeder that would use some form of pet identification. To have a successful system, we decided that the cat would need to come within a certain range of the feeder and meow in order to have the feeder dispense into the cat's dish bowl.

SCOPE

In scope

- Works with a wall outlet
- Voice recognition
- Will use a proximity sensor to determine proximity to the feeder
- Varying amounts of food available to dispense
- Simple to use
- Number of times that food can be dispensed can be controlled by the user

Out of Scope

- Camera/video record
- Wi-fi enabled application
- Wet food
    - Will only use dry food

Assumptions

We will assume that whenever the cat comes near the feeder during the correct time interval and meows then the feeder will dispense food into its dish.

Table 1: User Roles and Responsibilities

Role	Responsibility
Pet owner role	The pet owner will fill the food reservoir to the feeder machine. The pet owner will plug in the power cable for the feeder machine to the AC power.
Pet role	The pet will need to give the same sound that has been recorded and be in the vicinity of the feeder at the same time in order to get food from it.
Machine	To detect pet's voice, location and dispense the appropriate amount of food into its dish.

Pet Feeder with Pet Identification

MARKETING REQUIREMENTS

- Simple
- Accurate
- Robust
- Adjustable
- Light in Weight (Plexiglass)
    - To help with the weight requirement, we wanted to use plexiglass as the material to build the feeder with
- Efficient

SPECIFICATION REQUIREMENTS

- Varies the dispense times between feedings (3-5 hours)
- Three different volumes of food that can be dispensed (small, medium, and large)
- Arduino will be powered through a 12V wall adapter at 5V
- Recorded cat meow was broken into 5 separate voice segments and filtered to work with voice recognition software/hardware
- As part of a battery backup circuit, the 12V wall adapter will charge a 9V battery to be able to power the system in case of a power outage

Pet Feeder with Pet Identification

HIGH LEVEL DESIGN

System overview

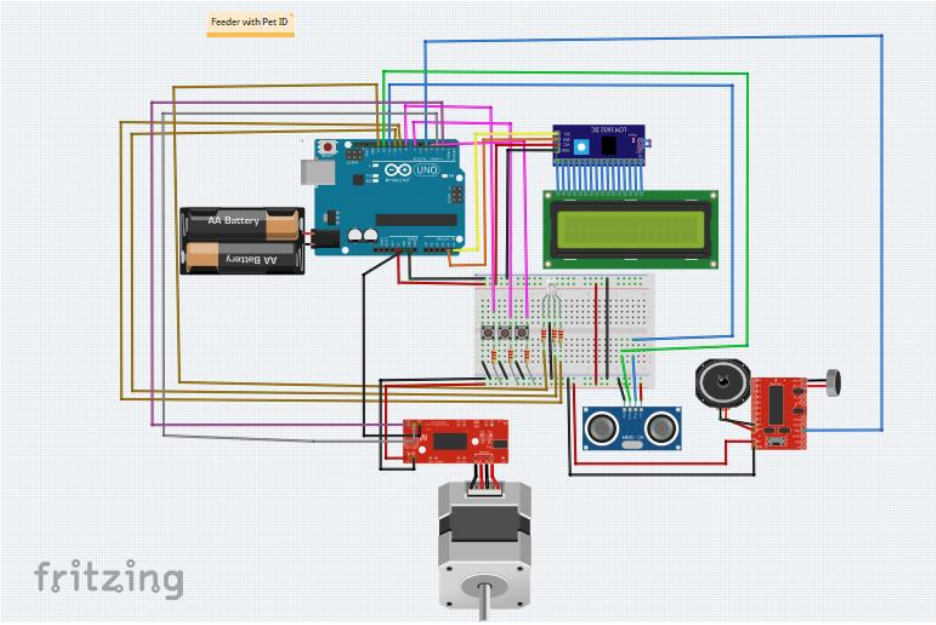


Figure 1 Displays the general system overview that was created when we first began the project

System-Wide Design Decisions

To make the decision of what components that will be used in the project, we have established a weighted value for each criterion. The criteria are based on the project's budget, in scope, customer requirements. These criteria are:

- Cost (3): it ranges from $20 to $60. If the price of the component is less than $20 then affected coefficient is two; less than $40 we give coefficient is one; above $60, the coefficient is zero
- Power Consumption (4) on output pin. Range from 0.5-4 Watts. If power consumed is less than 1 Watt then coefficient affect is two; if less than 2 Watts the coefficient is one; finally, if greater than 3 Watts then coefficient is zero
- Accuracy (4) is based on how accurate the chosen mechanism will dispense the cat food. If the mechanism dispenses the exact amount of food desired by the owner, then the coefficient is two. If it is food dispensed is close enough (+/- ? oz) to the right size, then the coefficient is one. The last case says that if the size can not be predicted at all, then the coefficient is zero
- Development resource or Interface (3) is based on the library availability as well as free and open resources

  - The coefficient is zero if no resource(s) are available, one if the resources are available. It will receive a two if resources are available and they are simple to install
- Durability (3) is based on how long the mechanism can run with no issue. If it is less than one year, the coefficient is zero, if less than two the coefficient is one, if more than two years, the coefficient is two
- Peripheral feature (3) deals with the external port availability. Boards have essential ports such as power, USB and adapters like I2C (serial communication chips). If board has less periphery features for the project, then the coefficient is zero, adequate peripheral coefficient is one, and if the selected board has more than the required peripheries, then coefficient is two
- Familiarity of the software and simplicity of hardware setting (2) is based on previous experience and the ability to learn. If at least two members of the team have some previous experience on the software and hardware setting, then the coefficient is two. If one member has experience, then the coefficient is one. If none of the team has experience on the software, the coefficient is zero. We all want to learn the software and hardware part of the project, but if no one has experience to help others, it will be difficult for us to come to common ground and understand each other.

Table 2: Choice of Microcontrollers

After searching a variety of microcontrollers characteristics, we have identified three microcontrollers that could be used for the cat feeder project. These microcontrollers are Raspberry Pi B, Arduino UNO, and Beaglebone Black. To select the microcontrollers that best fit our customer requirements and the budget assigned, we have established selection criteria.

Criteria	Weight	Raspberry Pi B (0-2)	Arduino UNO (0-2)	Beaglebone Black (0-2)
Price	3	1	2	0
Familiarity and Simplicity	2	1	2	1
Peripheral Features	3	2	1	2
Reliability	3	1	2	1
Platform Development	3	1	2	1
Power Consumption	4	1	2	1
Total Score		21	33	18

Looking at the totals, Arduino UNO ($8.49) was the right microcontroller for this project. It's priced less than the Raspberry PI B ($36.40) and the Beaglebone Black ($66.99). Secondly, we also focused on the power consumption. Once again, Arduino's (250mW) power consumption is less than Raspberry Pi B and Beaglebone which are approximatively (2W). In, addition to the above asset, Arduino has an easy setup over the Raspberry Pi which requires an additional resource like Wi-fi and a SD card. The availability of Arduino development software and libraries make the board simple to interface it with an analog sensor, motor, and other components. To add to these results, we ended using an Arduino Mega. As the project went on, we decided it would be best to up our microcontroller seeing as though we were plugging a few more components than we originally anticipated and Arduino Mega was the better option to use for this. It has more features and peripherals to work with.

Table 3: Dispense Mechanism

Criteria	Weight	Feed Screw	Paddle	Sliding Door
Precision	4	2	1	1
Reliability	3	1	1	1
Durability	3	1	1	1
Total		16	10	10

After further consideration, we ended up using the sliding door as the appropriate the dispense mechanism. We decided that a feed screw may not be the best mechanism after meeting with our customer. Our customer was worried about potential food getting stuck in the screw and it not being able to deliver the appropriate amount of food each time.

Pet Feeder with Pet Identification

Concept of Execution

In the following diagrams, we have our original execution plans. Things changed as the project progressed on but are placed in this document to show what we originally planned to do before changes were made and implemented.

Original Architectural Design

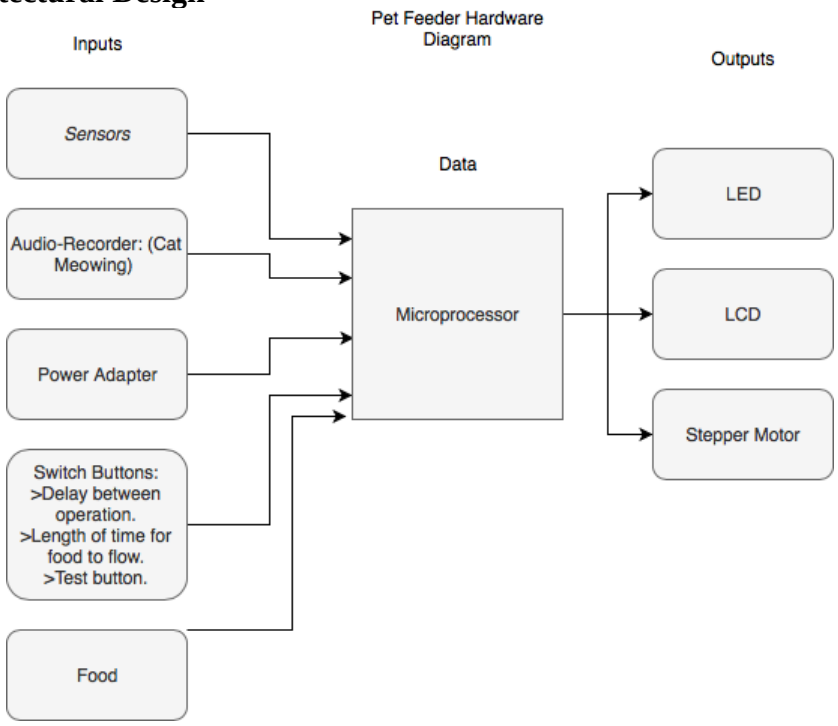


Figure 2 Displays our original hardware diagram for the feeder. There were a few changes made after this orginal diagram was released.

Pet Feeder with Pet Identification

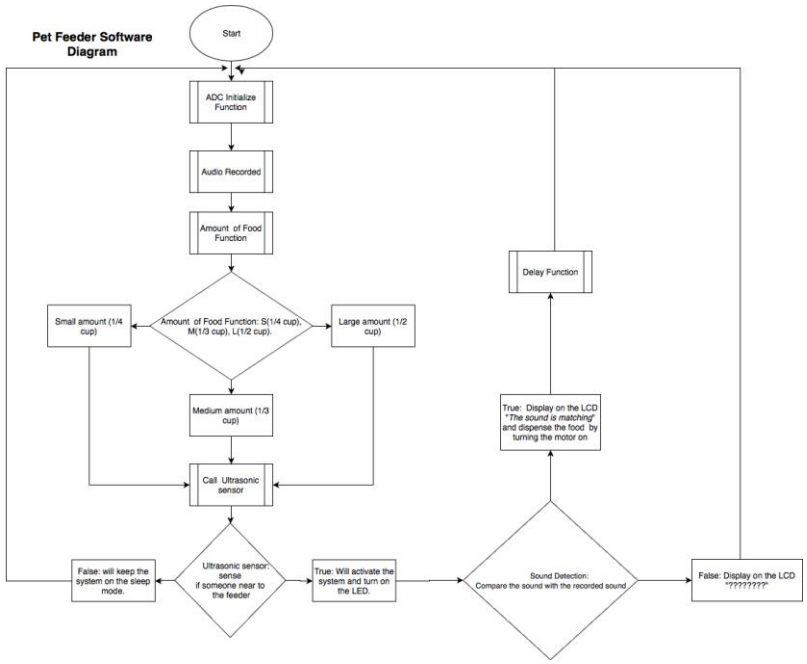


Figure 3 Displays the original software flowchart for the system. Once again, this flowchart changed later on in the project and is reflected in the Appendixes section

Interface Design
The user will see the following inputs and outputs on the machine

Inputs:

- Four push buttons that allow the user to operate the feeder
  - The first is used to set the food size
  - The second is used to set the time interval
  - The third is used as a test button
  - A fourth push button that allows the system to be reset
- Ultrasonic sensor to detect proximity
- Microphone to pick up cat's meow
- Food reservoir to place food in
- Power adapter that'll convert AC to DC

Outputs:

- LCD will display what option(s) user selects
- LED will operate as the feeder is working

LOW LEVEL DESIGN

Table 4: Programming Language

Criteria	Weight	C (0-2)	C# (0-2)	C++ (0-2)	Java (0-2)	Python (0-2)
Familiarity	2	1	1	1	1	1
Platform Restrictions	3	2	1	2	1	1
Total		7	5	7	5	5

Table 5: Choice of Motor/Solenoid

Criteria	Weight	Stepper Motor (0-2)	Servo Motor (0-2)	Solenoid (0-2)
Price	3	2	2	1
Power	3	1	2	1
Speed / Loop	3	1	2	1
Total		12	18	9

Table 6: Choice of Motor/Solenoid Details

Motor/solenoid	Speed / response time	Cost	Max Power
Servo Motor SG90	3000 RPM	$ 3.00	400mA * 5V = 2W
Stepper Motor	1200 RPM	$ 3.91	1.3A * 3.25V = 4.2W
Direct Mini Push-Pull Solenoid	30ms	$ 9.99	5V * 830mA = 4.15W

Pet Feeder with Pet Identification

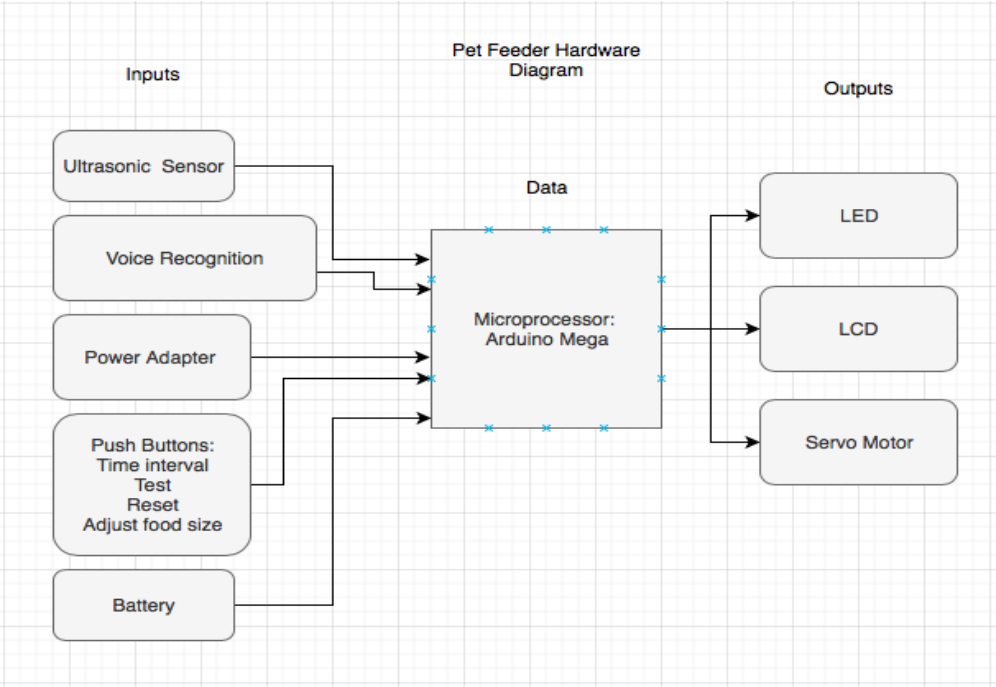


Figure 4 Shows an updated Hardware Diagram for the feeder

Detailed Hardware Design

1. Microprocessor:

Figure 5 Displays the Arduino Mega

The Arduino Mega microprocessor will be used to control the outputs depending on the result of the inputs. The components of the systems will be connected to the Arduino using wires and a printed circuit board.

2. Push buttons:

Figure 6 Displays the push buttons that we used

There are four push button inputs connected to the Arduino Mega. The push buttons were used to select the food size, time interval, test, and to reset.

3. Power Adapter:

Figure 7 Displays the power adapter

The power adapter will be converting the voltage from AC to DC 12V.

4. Ultrasonic Sensor:

Figure 8 Displays the HC-SR04 Module

The sensor will be used to detect the proximity of the cat. We used the HC-SR04 module.

5. Voice Recognition Module:

Figure 9 Displays the voice recognition module in which we used

The voice recognition module will be used to compare the current sound of the cat with the recorded sound to dispense food from the feeder. The voice recognition that we chose for our project is the Geeetech Voice Recognition Module.

6. LEDs:

Figure 10 Displays LEDs

LEDs will turn on when the dispenser opens to dispense the food and close.

7. LCD & I2C:

Figure 11 Displays the 20x4 LCD and I2C

LCD will display the result of an action such as when the user selects a food size it will display on the screen the size that was selected. The LCD that we chose for our project 20x4 2004A LCD display with I2C.

8. Motor:

Figure 12 Displays the servo motor we used

The motor will be used to dispense the food from the machine. We chose a servo motor instead of other motors based on previous research.

9. Real Time Clock

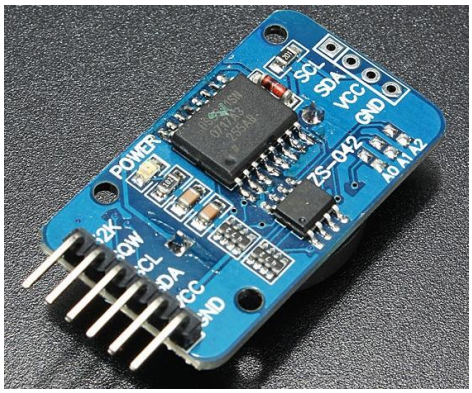


Figure 13 Displays the real time clock

The real time clock was added in the latter half of the project. This clock was tied in with the LCD screen to keep an accurate account of the time and to help the feeder keep track of time between feedings.

Pseudocode:

Program Feeder with pet ID

Pet Feeder with Pet Identification

ADC Initialize function

Audio Recorded: will record the sound and converted to digital value

Switch cases (Amount of food)

Case1: Small amount

LCD: Display on the LCD “Small amount”

Break;

Case2: Medium amount

LCD: Display on the LCD “Medium amount”

Break;

Case3: Large amount

LCD: Display on the LCD “Large amount”

Break;

Default:

END Switch cases

Dispense the food when the user pushes the button

Call Ultrasonic Sensor

While (Sensor)

If something is near the feeder it will wake up and turn on a green LED

Else

The machine remains in sleep mode
End if

END While

Microphone: will get the current sound and convert it to digital

While (Compare function)

If the current sound equals the recorded sound, a message will show on the LCD “Sounds Match”, and the motor will turn opening the sliding door to dispense the food

If the user selected a small amount of food to dispense, the door will open for three seconds, and close after

If the user selected a medium amount of food to dispense, the door will open for six seconds, and close after

If the user selected a large amount of food to dispense, hold the door open for nine seconds, and close after

Display on the LCD “Sound Match”

Else

Display on the LCD “Sounds Don’t Match”

Return to the beginning of the program

End if

END While

Delay Timer Function

LCD: Display on the LCD the delay time that has been selected by the user

Return to the beginning of the program

TEST METHODOLOGY & TEST RESULTS

Test 1 *Arduino Power*

Purpose: The purpose of this first test is to determine whether the Arduino is getting the needed power to run the system

Materials Needed: Arduino Mega, Printed Circuit Board, Adapter/or USB connector, Computer, Ohmmeter

Procedure:

1. Using a wall adapter, plug it into the wall. Make sure all wires are connected to the circuit board
2. After establishing power from the wall, a green light should illuminate from the Arduino board
3. If this light is on, then we will then use an ohmmeter to measure the output voltage we are getting. We want to run the Arduino at 5V
4. If the green light isn’t illuminated check your wiring connections and power source
5. Using a USB connector, then you would plug one end into the computer and the other end into the Arduino. The same green light should illuminate

Test 2 *Ultrasonic Sensor*

Purpose: The objective of this test it to determine whether the sensor can detect a proximity, more importantly the cat’s proximity to the feeder

Pet Feeder with Pet Identification

Materials Needed: Arduino Mega, Printed Circuit Board, Adapter/or USB connector, Computer, Ultrasonic Sensor

Procedure:

1. All we would need to do to test this sensor is to place an object near it to see what distance is read. The resulting distance should display on the LCD screen

Test 3 *Voice Recognition*

Purpose: Objective of the voice recognition to record/listen for the cat's meow to be able to dispense food

Materials Needed: Arduino Mega, Printed Circuit Board, Adapter/or USB connector, Computer, Voice Recognition Module

Procedure:

1. With the VR module plugged into the Arduino Mega and the circuit board, we will playback the recorded meow from Lucas to test it
2. LCD should display a message that reads, "cat's voice ok"

Test 4 *Servo Motor*

Purpose: The objective of testing the servo motor is to be able to open/close the door that allows cat food to flow out

Materials Needed: Arduino Mega, Printed Circuit Board, Adapter/or USB connector, Computer, Servo Motor

Procedure:

1. With the servo motor connected to the Arduino and the circuit board, we will run the program to test the opening and closing of the door that will allow the cat's food to dispense and to also ensure that the servo motor works properly
2. We will first run a minimum of 10 trials to ensure the motor has full functionality. We are wanting the motor to rotate a full revolution clockwise. Then we want to do the same going counterclockwise. While doing this we should also see the sliding door operating alongside the motor to fully open and close
3. Each volume of food will be tested to help verify that the servo motor will operate for the appropriate of time
4. Will run test a minimum of 10 times each for each volume size of food

Test 5 *Software*

Pet Feeder with Pet Identification

Purpose: The purpose of testing the software the pet feeder is to determine whether the entire code will allow the system to function properly

Materials Needed: Arduino Mega, Printed Circuit Board, Adapter/or USB connector, Computer

Procedure:

1. Now that we have tested each component separately, we will now test the software by itself to see if we can output some results
2. We will test each section of the code for proper functioning
    a. Time delay
    b. Volume of food being dispensed
    c. Sleep mode (sleep/wake up)

Test 6 *Feeder*

Purpose: The purpose of testing the entire is to determine whether our entire system/project is working. This is the big test. We are testing the entire feeder as one full unit

Materials Needed: Arduino Mega, Printed Circuit Board, Adapter/or USB connector, Computer, Ohmmeter, Servo Motor, Voice Recognition Module

Procedure:

1. We will first test the feeder using the smallest settings that we originally agreed upon
    a. For example, the smallest setting has a delay time of 3 hours, the smallest volume of food
    b. The next size up would have a delay time of 4 hours, and have the medium volume of food dispensed
    c. The last size would have a delay time of 5 hours, and have the largest volume of food dispensed
    d. These settings are interchangeable and will be tested as well. For example, my cat eats every 3 hours, but will eat the largest
    e. volume of food to dispense

Pet Feeder with Pet Identification

Engineering Requirements	Test Necessary
Varying dispense times	Yes
Volume of food being dispensed	Yes
Arduino powered at 5V	Yes
Battery being charged when not in use	Yes
Sensor detects 25 cm or less of feeder	Yes

Results		
Test	**Accepted Criteria**	**Pass/Fail**
Arduino	Powers at 5V Green light illuminates Ohmmeter reads 5V	Yes
Ultrasonic Sensor	Sensing range 2-100cm, ideally, we want 25cm (10 in)	Yes
Voice Recognition	Pickup cat's meow Able to compare current meow with recorded one	Yes
Servo Motor	Fully operate by opening/closing the door Rotate +180°/-180°	Yes
Software	Able to switch between functions to operate feeder Delay functions work properly	Yes
Feeder	Able to dispense different volumes of food that include small, medium, and large	No

From the results table you can see that everything passed the test that we ran except for the actual feeder. The feeder didn't dispense the proper food sizes. Through testing we found that the feeder dispense wasn't very accurate. For example, we were able to run the small size of food and received the same amount of food from the feeder each time. However, when we began to increase the food size the amounts weren't consistent. One test in which we ran using the medium food size, the feeder dispensed a half a cup of cat food. To ensure that was an appropriate amount we ran the test again using the same settings and the dispenser pushed out more than a half a cup.

CONCLUSIONS, RECOMMENDATIONS, & INDIVIDUAL ASSESSMENT

After all the testing and dispensing, we came up with some conclusions and recommendations that would help us in the future if we wanted to perfect our feeder. One conclusion that we made dealt with the physical dispenser itself. We believed that the mechanics of that component needed to be changed. We thought that the slide should have been angled more to prevent any food from not being dispensed into the cat's dish. Another conclusion that we came up with dealt with the servo motor. It was found through addition research that when servo motor operates (open and close), they don't usually return to their true home position. So, maybe another motor would be a better option to think about. The last conclusion that we made also dealt with the servo motor since it was what gave the most trouble. To make it more accurate, we thought to create a function just for the servo motor to delay its operating time for each food size.

REFERENCES

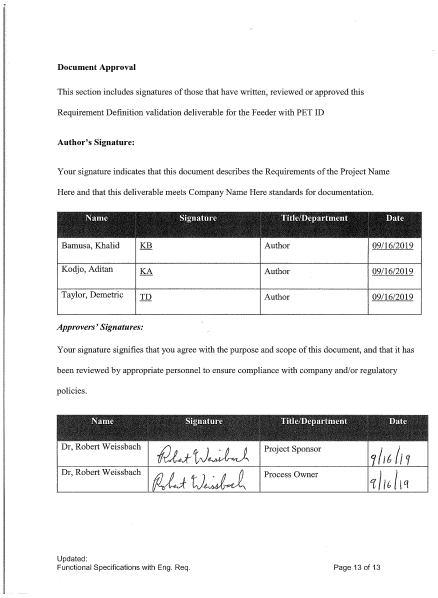
Title	Source	Comment
Raspberry Pi or Arduino Uno? One Simple Rule to Choose the Right Board	https://makezine.com/2015/12/04/admittedly-simplistic-guide-raspberry-pi-vs-arduino/	Allows us to understand Arduino and Raspberry Pi features
Stepper & Servo	https://www.islproducts.com/designnote/stepper-motors-vs-servo-motors/	Allows us to develop knowledge of which motor will fit our design
Raspberry Pi 3 Vs Arduino Uno Rev3	https://www.arrow.com/en/research-and-events/articles/comparing-arduino-uno-and-raspberry-pi-3	Better understand the advantages and disadvantages of both boards
Power Consumption Benchmarks	https://www.pidramble.com/wiki/benchmarks/power-consumption	Understand Raspberry Pi type and its power consumption
Arduino Ultrasonic Sensors	http://www.f15ijp.com/2012/09/arduino-ultrasonic-sensor-hc-sr04-or-hy-srf05/	HC-SR04 & HY-SRF05 difference
Proximity Sensor Selection Guide	https://cdn.automationdirect.com/static/specs/proxselection.pdf	To learn sensor specifications and their sensing envelopes

Pet Feeder with Pet Identification

Title	Source	Comment
Electrical Safety	NFPA 70E	No exposed wires, equipment, fuses, etc.
Determination for Center of Gravity (CG)	ISO 10392:2011	Use horizontal planes to determine CG
Accuracy (trueness and precision)	ISO 5725-1	Determination of repeatability, reproducibility

This link is the circuit in which we found for our battery backup
https://www.mycircuits9.com/2012/07/simple-battery-backup-circuit.html

APPENDIXES



Document Approval

This section includes signatures of those that have written, reviewed or approved this Requirement Definition validation deliverable for the Feeder with PET ID

Author's Signature:

Your signature indicates that this document describes the Requirements of the Project Name Here and that this deliverable meets Company Name Here standards for documentation.

Name	Signature	Title/Department	Date
Bamusa, Khalid	KB	Author	09/16/2019
Kodjo, Aditan	KA	Author	09/16/2019
Taylor, Demetric	TD	Author	09/16/2019

Approvers' Signatures:

Your signature signifies that you agree with the purpose and scope of this document, and that it has been reviewed by appropriate personnel to ensure compliance with company and/or regulatory policies.

Name	Signature	Title/Department	Date
Dr, Robert Weissbach	Robert Weissbach	Project Sponsor	9/16/19
Dr, Robert Weissbach	Robert Weissbach	Process Owner	9/16/19

Updated:
Functional Specifications with Eng. Req. Page 13 of 13

Figure 1 Shows our Signed Functional Specifications

Pet Feeder with Pet Identification

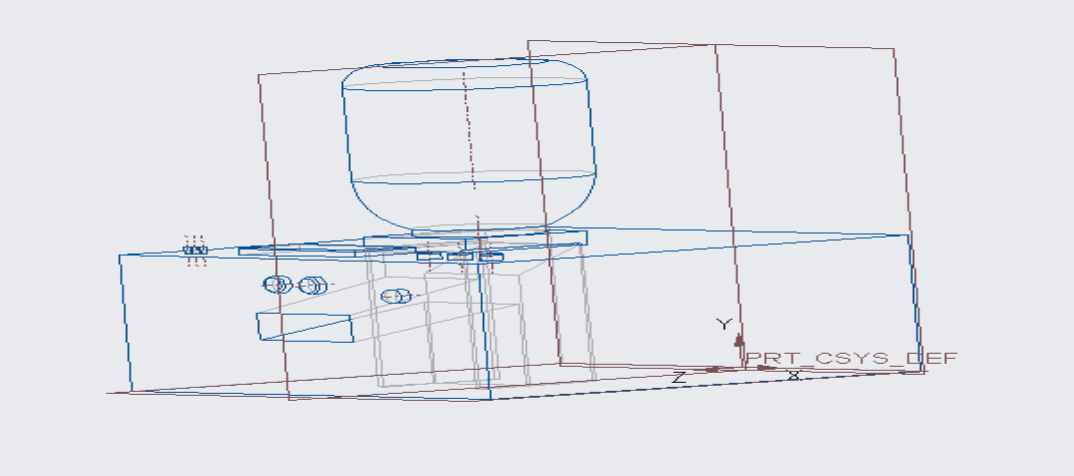


Figure 2 Displays the pet feeder design

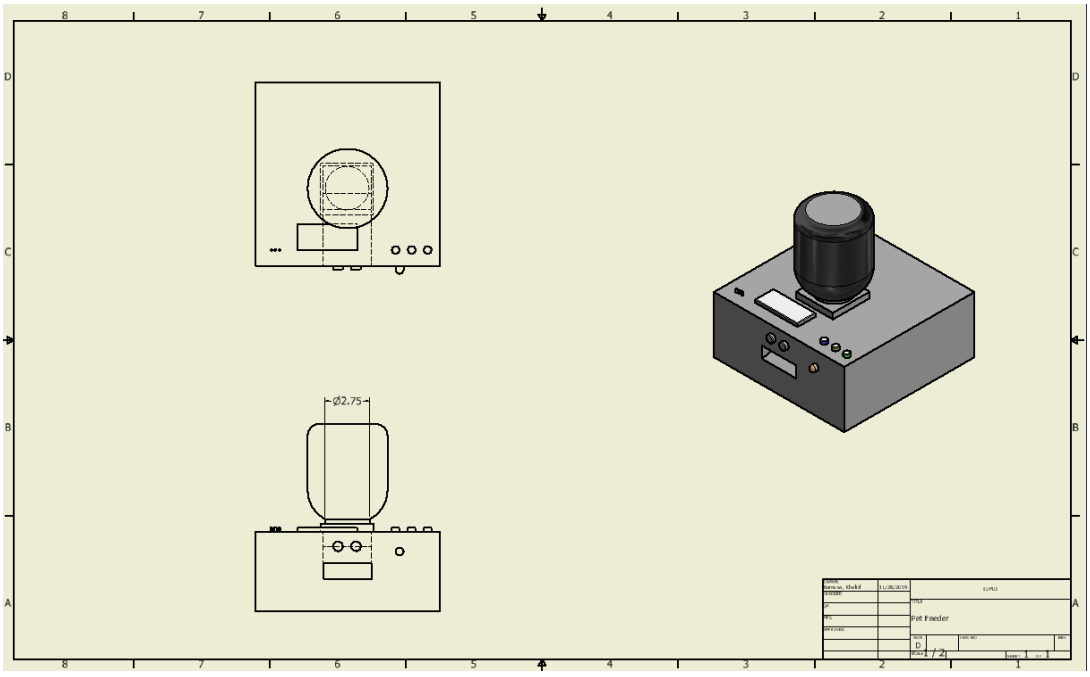


Figure 3 Displays the feeder design with dimensions

Pet Feeder with Pet Identification

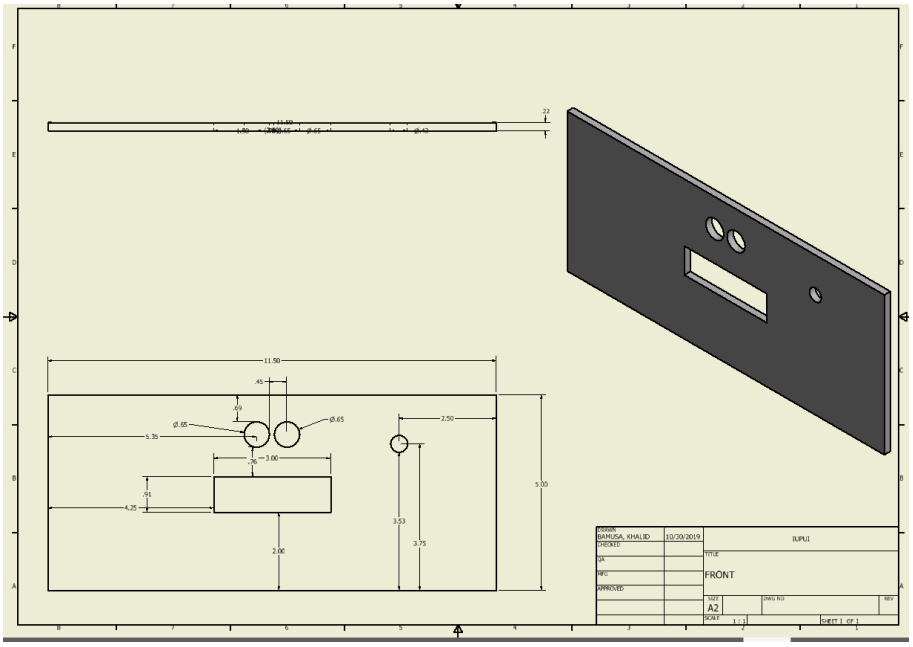


Figure 4 Displays the front panel of the feeder

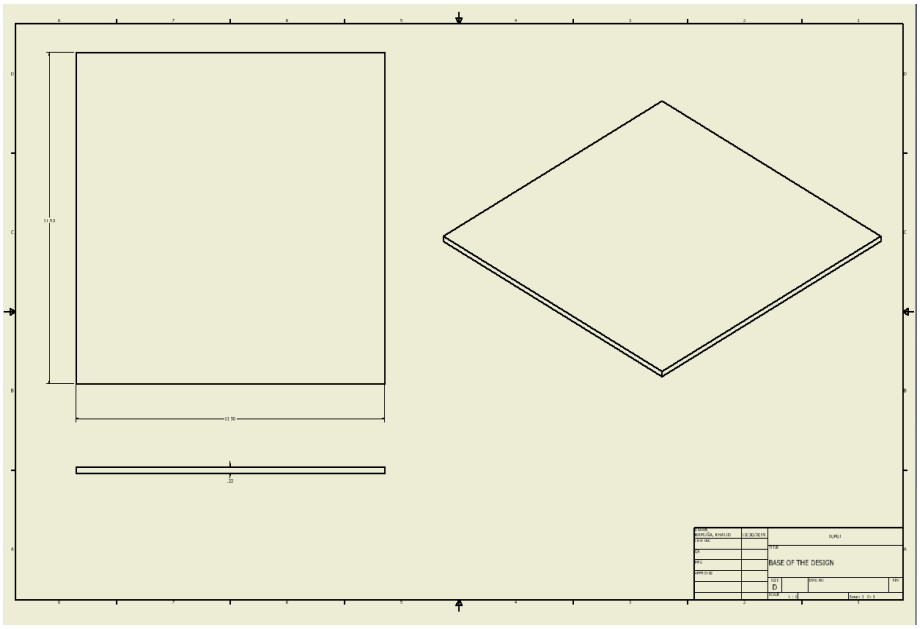


Figure 5 Displays the base panel of the feeder

Pet Feeder with Pet Identification

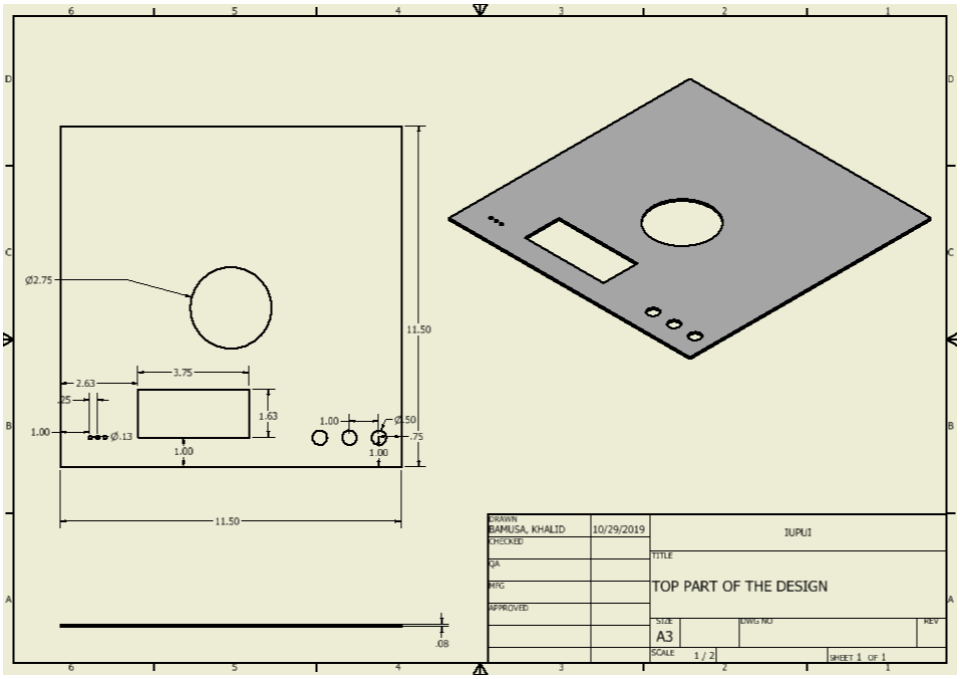


Figure 6 Displays the top panel of the feeder

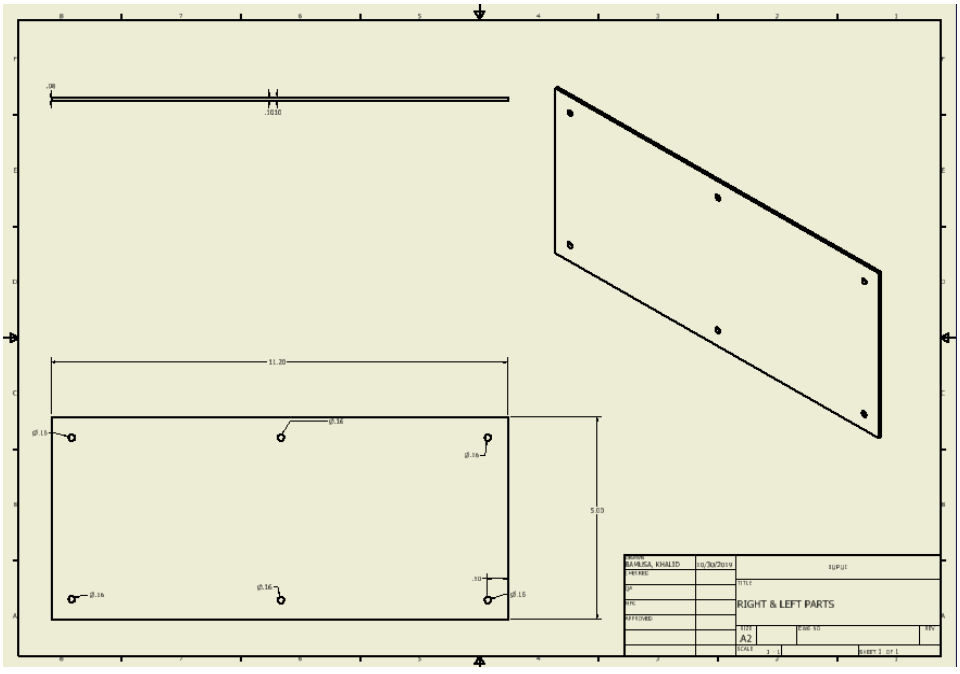


Figure 7 Displays the right and left panels of the feeder

Pet Feeder with Pet Identification

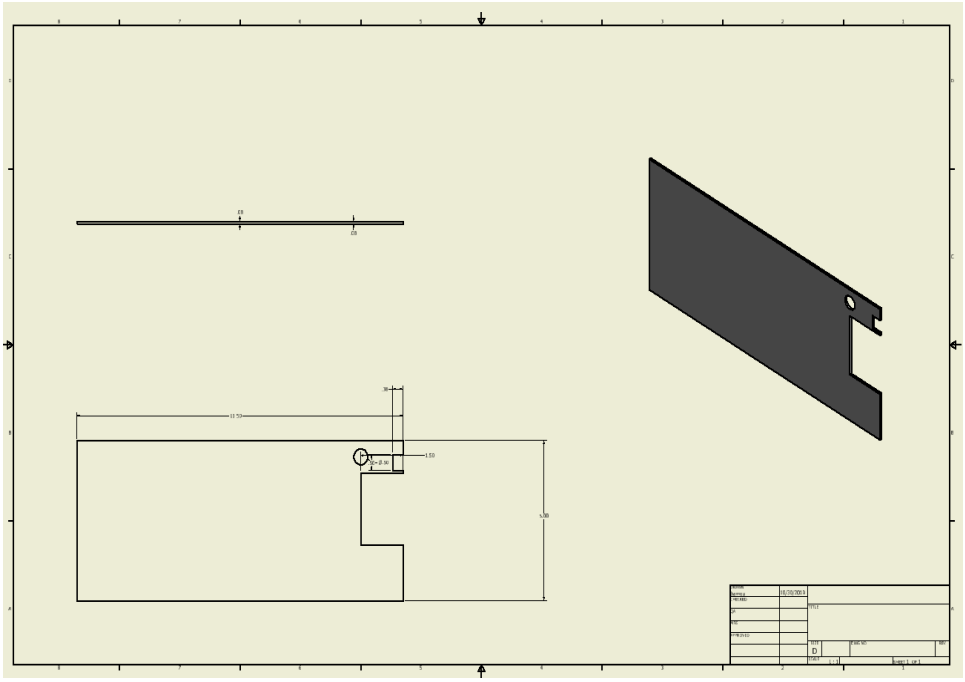


Figure 8 Displays the back panel of the feeder

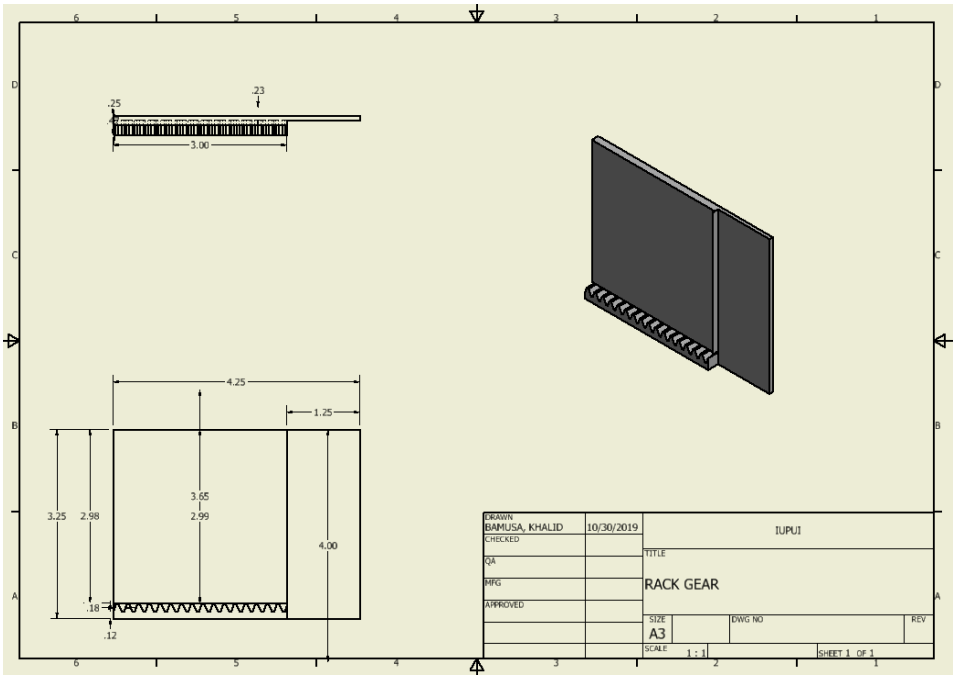


Figure 9 Displays the sliding door that opens and closes allowing food to dispense

Pet Feeder with Pet Identification

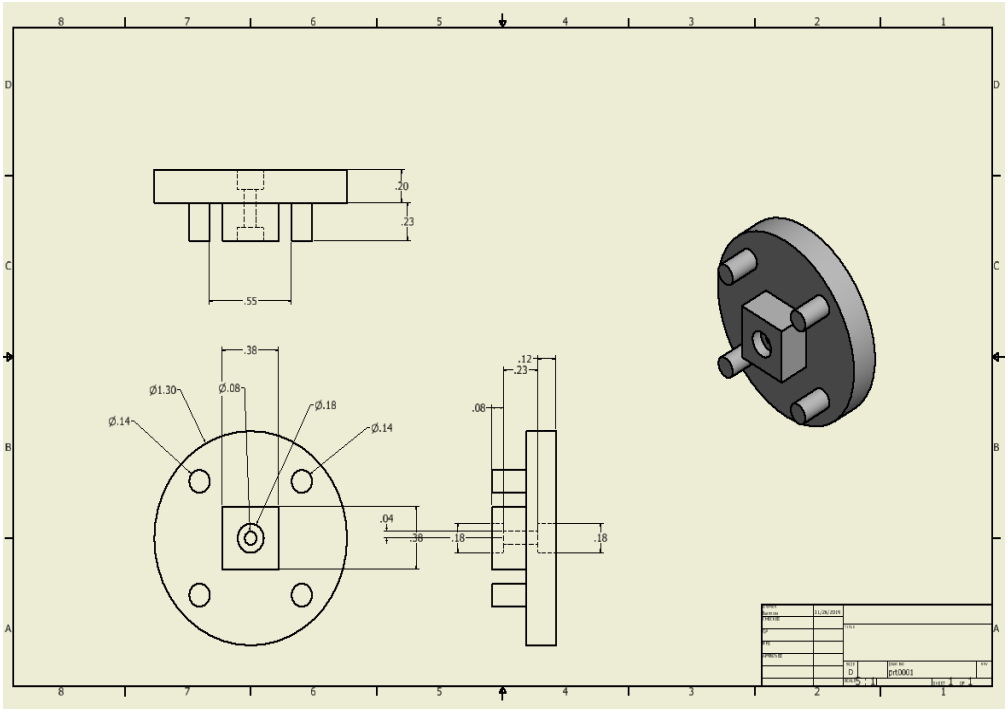


Figure 10 Displays the connector between the gear and the servo motor

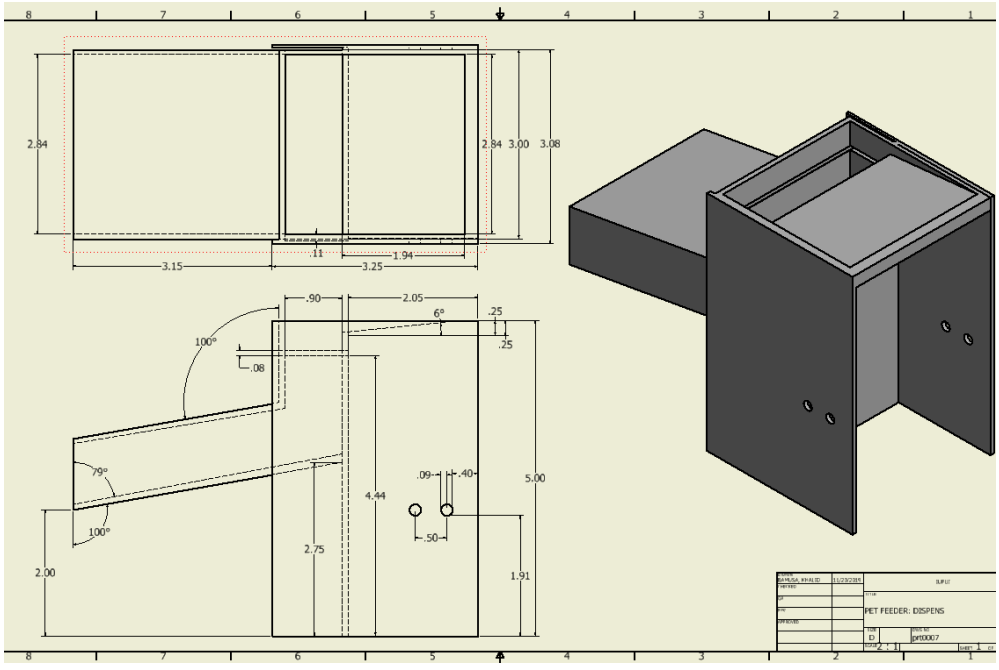


Figure 11 Displays the dispenser of the feeder

Pet Feeder with Pet Identification

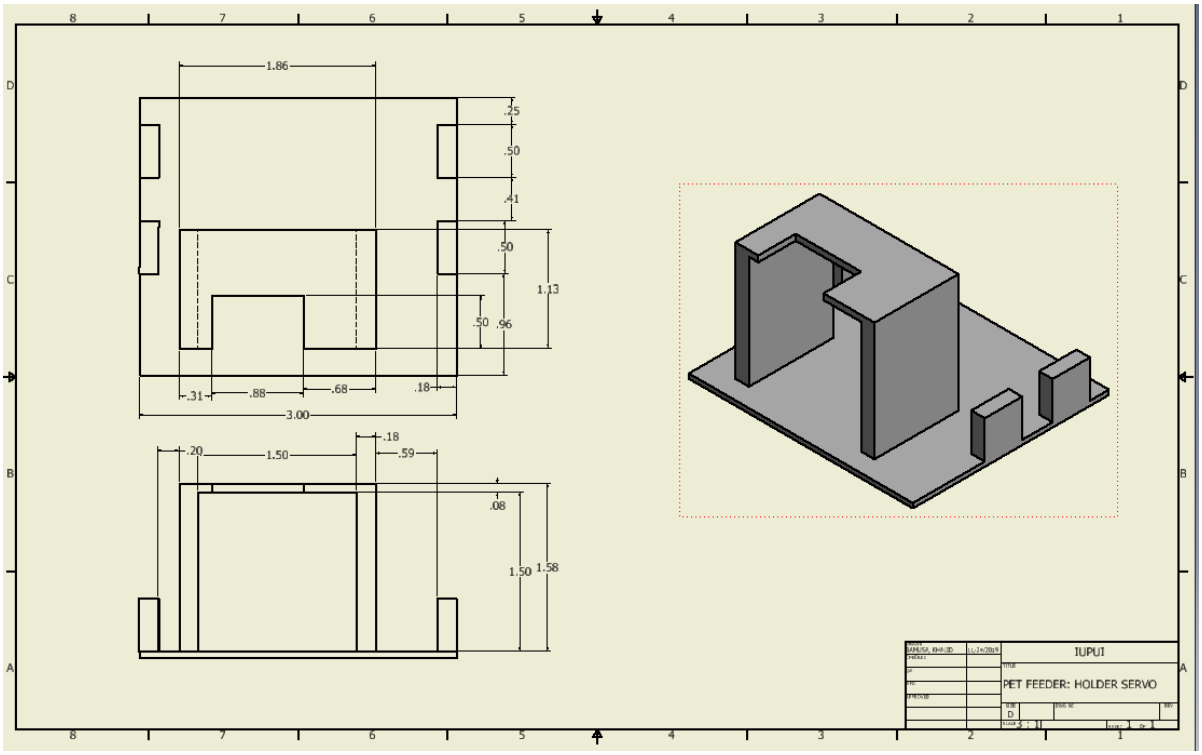


Figure 12 Displays the servo motor holder

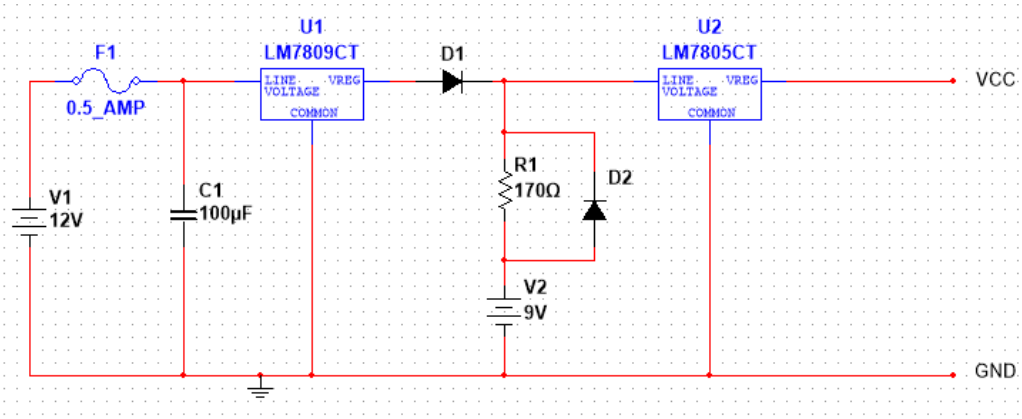


Figure 13 Displays our battery backup circuit schematic

Pet Feeder with Pet Identification

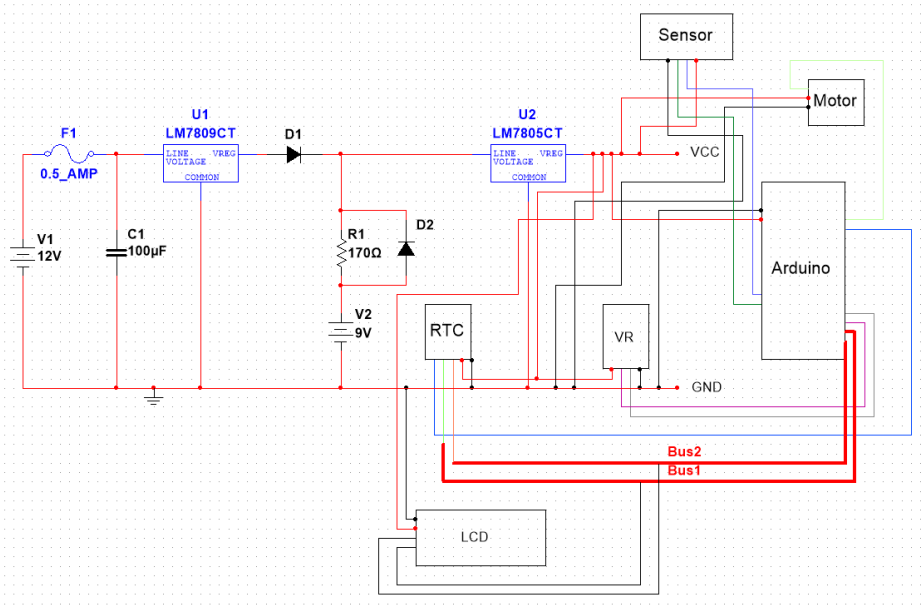


Figure 14 Displays our major components circuit

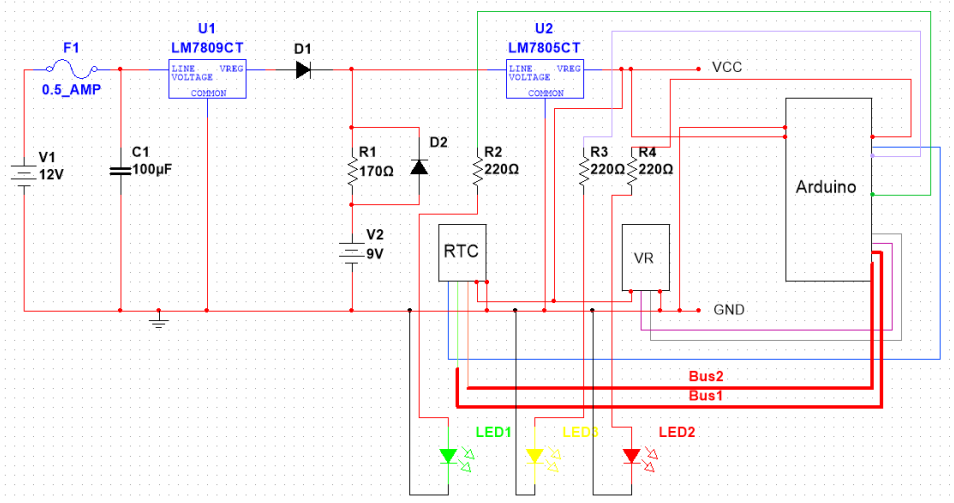


Figure 15 Displays the layout of our printed circuit board

Pet Feeder with Pet Identification

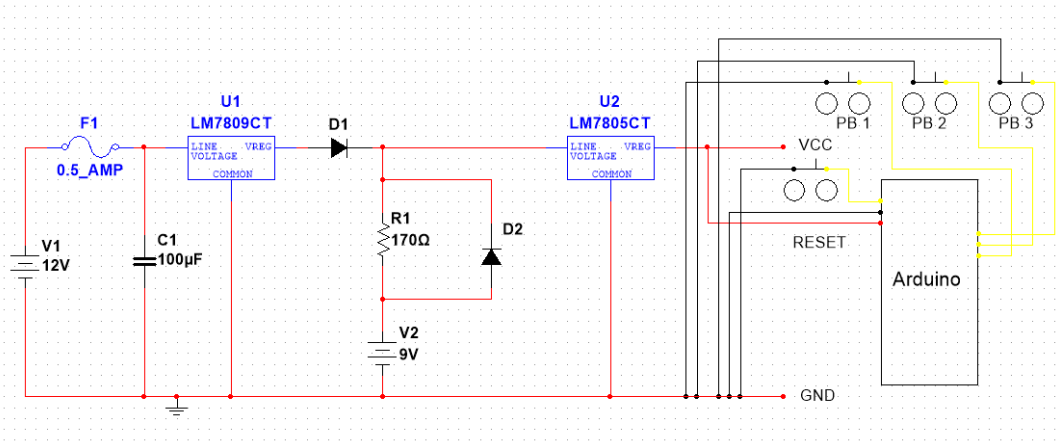
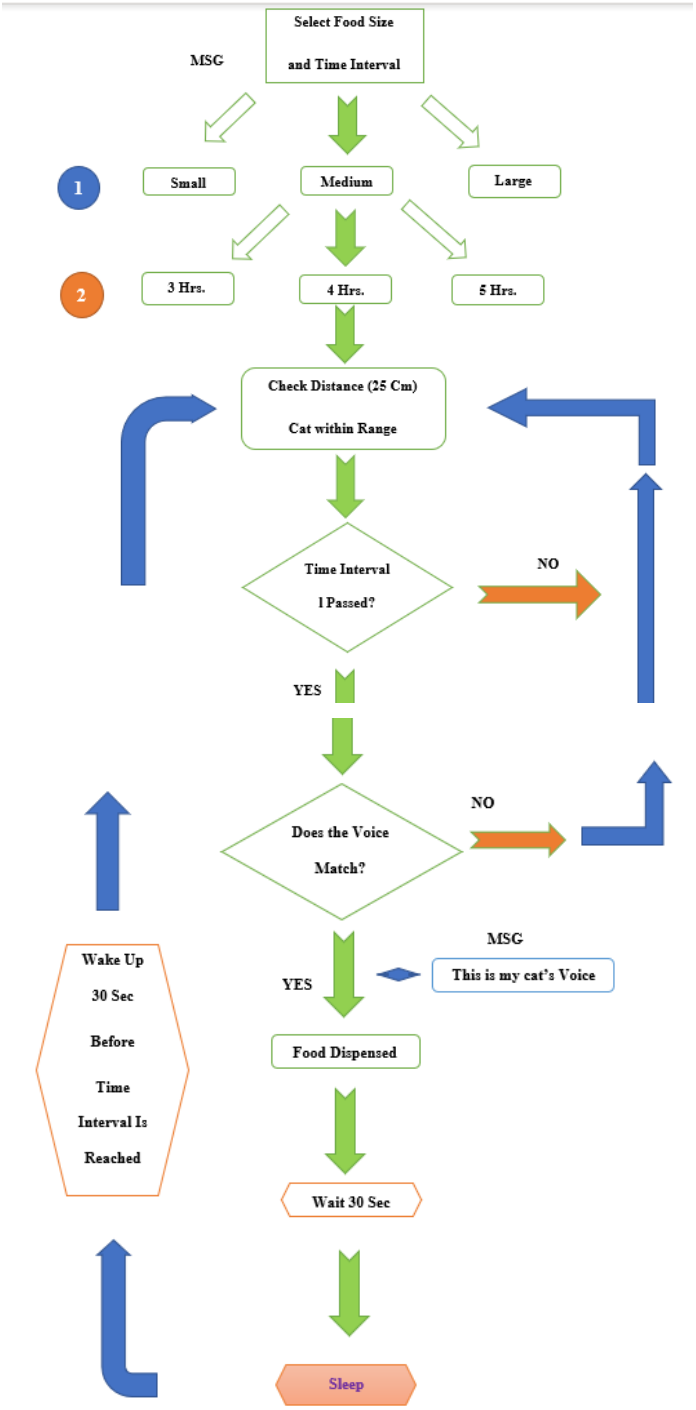


Figure 16 Displays our push buttons in our circuit

Pet Feeder with Pet Identification



Figures 17 and 18 Display the flowchart of feeder food dispense process

Pet Feeder Operation Manual

Start the Feeder

1. **Connect power cable**

   When the feeder starts the red LED should be illuminated

2. **Select food size and time interval**

   Press blue push button twice and yellow button twice

   This will select small size and time interval three hours

   The size and time can be increased when continue pressing these buttons

3. **Reset food size and time interval**

   Press blue push button five times and it will reset the food size to default

   Press yellow push button five times and it will reset the time interval to default

   This activates the default feeding

Default Function

It is a default feeding that allows the feeder to dispense food without requiring the voice of the cat. When the user plugs in the machine for the first time, it requires the user to press the blue and the yellow push button one time to be activated

Factory Reset

This clears the feeder food size and time interval in the EEPROM memory. To factory reset your feeder, you will need to do the following;

1. Press and hold the blue push button until the counter increments and displays a

   reset message displays on the LCD screen

2. Press and hold the yellow push button until the counter increments and displays a

   reset message displays on the LCD screen

Pet Feeder with Pet Identification

3. Press and hold the green push button until the counter increments and displays a reset message displays on the LCD screen
4. A message is displayed on the LCD screen asking to cycle power the device
5. When the feeder resumes power, a massage will display on the screen to choose food size and time interval

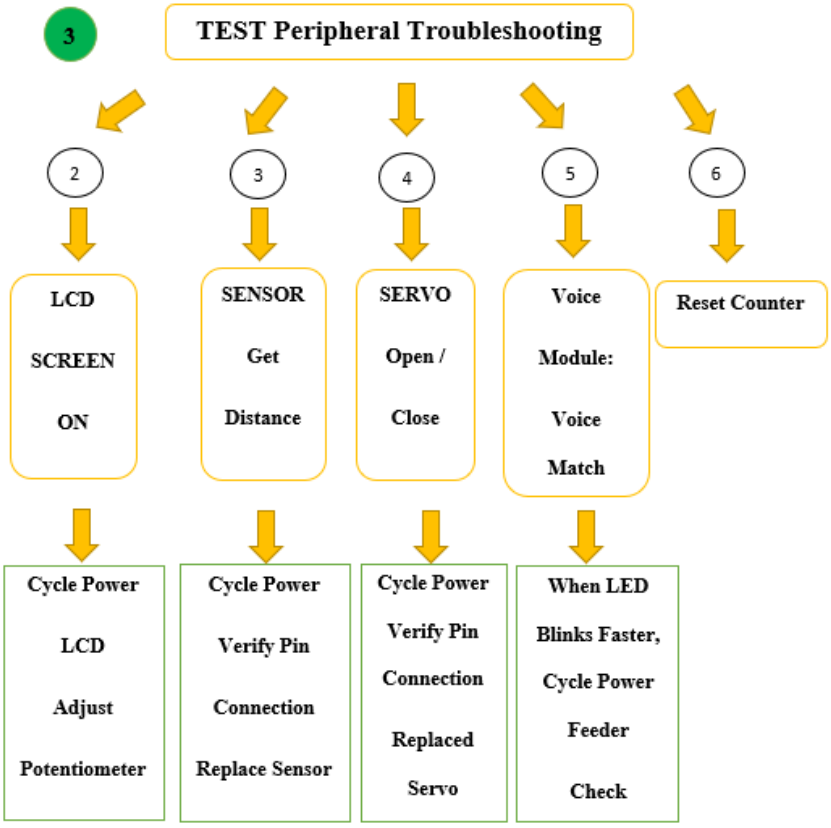
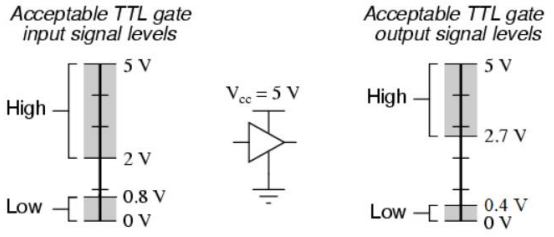


Figure 19 Displays the Peripheral Troubleshooting Test

Geeetech Voice Recognition Specification

Pet Feeder with Pet Identification

- 15 voice instructions gathered into 3 groups
- Voltage: 4.5 - 5.5V
- Current: < 40mA. Low power consumption
- Analog interface: 3.5mm mono-channel microphone connector + microphone pin interface
- Recognition accuracy: 99% (under ideal environment)
- Size: 30mm x 47.5mm. Allows the module to install into other electronic devices
- Digital Interface: 5V TTL level UART interface
- The internal circuitry of the voice recognition is built according to engineer requirements. The module respect TTL logic level standard is built form bipolar transistors to fulfill switching and maintaining logic states
    1. V_{OH} -- Minimum OUTPUT Voltage level a TTL device will provide for a HIGH signal is 2.7 V
    2. V_{IH} -- Minimum INPUT Voltage level to be considered a HIGH is 2.0V
    3. V_{OL} -- Maximum OUTPUT Voltage level a device will provide for a LOW signal is 0.4V
    4. V_{IL} -- Maximum INPUT Voltage level to still be considered a LOW is 0.8V



Source: https://learn.sparkfun.com/tutorials/logic-levels/ttl-logic-levels

- The serial data format: 8 data bits, no parity
- 1 stop bit
- The default baud rate is 9600 and can be changed

Pet Feeder with Pet Identification

	Design	142 days	Fri 3/29/19	Sun 10/13/19		
	Research effect of the sensor frequency on the cat	11 days	Mon 5/6/19	Mon 5/20/19	Aditan Kodjo	Compare frequencies between the sensor and what a cat hears
	Order Voice Recognition Hardware	29 days	Mon 5/13/19	Thu 6/20/19	Aditan Kodjo	Purchase hardware that will be used to detect cat's voice
	Program and Test Voice Recognition Module	15 days	Fri 6/21/19	Thu 7/11/19	Aditan Kodjo	Filtering, separating, etc.
	Build a prototype of a working system	37 days	Fri 6/7/19	Mon 7/29/19	Aditan Kodjo, Khalid Bamusa,De	Replica of what the finished feeder will look like
	Wiring Diagram	23 days	Sun 6/30/19	Tue 7/30/19	Demetric Taylor	Wiring of entire system from start to finish
	Circuit Schematic	69 days	Sun 6/30/19	Wed 10/2/19	Demetric Taylor	Schematic that shows exactly how each component is connected to the next
	Update Gantt Chart	83 days	Mon 6/10/19	Wed 10/2/19	Demetric Taylor	Current tasks and projected tasks for phase 2
	User Instructions	91 days	Mon 6/10/19	Sat 10/12/19	Khalid Bamusa	How the cat and owner will interact with the system
	Have 3D case printed	19 days	Wed 9/18/19	Sat 10/12/19	Khalid Bamusa	After approval from Dr. Weissbach and Steven
	Physical layout and servo motor construction	31 days	Mon 6/10/19	Mon 7/22/19	Khalid Bamusa	Shows where the servo motor will be placed inside the feeder
	Prepare to re-present phase 1	13 days	Mon 7/15/19	Wed 7/31/19	Aditan Kodjo,Kha	Rehearse for presentation
	Circuit Board	4 days	Sat 9/28/19	Wed 10/2/19	Demetric Taylor	Start and show to Dr. Weissbach
	Printed Circuit Board	19 days	Wed 9/18/19	Sat 10/12/19	Demetric Taylor	Get circuit board printed from Craig
	Acquire materials for construction	34 days	Wed 8/28/19	Sat 10/12/19	Aditan Kodjo, Khalid Bamusa,De	Send in request for remaining materials to get
	Construct feeder	12 days	Sun 10/20/19	Sun 11/3/19	Aditan Kodjo,Kha	Use time wisely to begin construction of the feeder. Use whatever resources needed
	Communicate	79 days	Wed 8/28/19	Mon 12/16/19	Aditan Kodjo,Der	THIS IS A MUST! We should all be communicating as much as possible to work on the project
	Reports	79 days	Wed 8/28/19	Mon 12/16/19	Aditan Kodjo,Der	Reports should be started sooner than later. Should be started the same week they are assignment
	Verify	**79 days**	**Wed 8/28/19**	**Mon 12/16/19**		
	Meet with advisor/sponsor	79 days	Wed 8/28/19	Mon 12/16/19	Aditan Kodjo,Der	Giver updates on the progress of the project and setup a meeting time(s)
	Weekly group meetings	77 days	Mon 9/2/19	Tue 12/17/19	Aditan Kodjo,Der	The time needed to really work together
	Find a battery solution backup to power the system	4 days	Wed 9/11/19	Mon 9/16/19	Khalid Bamusa	Battery backup
	Sleep mode on Arduino	26 days	Wed 9/4/19	Wed 10/9/19	Aditan Kodjo	Waking up the Arduino is the challenge to conquer
	Test/troubleshoot	11 days	Wed 10/9/19	Wed 10/23/19	Aditan Kodjo,Der	Test every component by itself before testing the whole system
	Communicate with each other on current progress of progress	4 days	Wed 10/23/19	Sat 10/26/19	Aditan Kodjo, Demetric Taylor,K	
	Final design	9 days	Sun 9/22/19	Wed 10/2/19	Aditan Kodjo,Der	Finalized paper design
	Final review	39 days	Tue 10/8/19	Fri 11/29/19	Aditan Kodjo,Der	Poster board for presentation, all assignments turned in, and review entire project
	Present final project	11 days	Mon 12/2/19	Mon 12/16/19	Aditan Kodjo,Der	Present the final project to those in attendance

Figures 16 and 17 Display our Gantt Chart for this project

Pet Feeder with Pet Identification

So, our budget was set at $250. The Bill of Materials will reflect that we stayed within that budget, but we actually surpassed it. We ended up spending over $300 total for the full project.

Bill of Materials

QTY	Part Name	Store	Part Number	Price
1	Felt Pads	Walmart	007452300729	$1.4
1	Brass RH WS 2*3.8 12CT	Lowe's	57387	$1.28
1	Acrylic Clear	Lowe's	11288	$33
1	Clr Acr	Lowe's	78778	$3
1	Machine Screw Nut 8/32	Home Depot	887480002522	$4.67
1	Phillips-Drive Machine Screws	Lowe's	755524	$3.96
2	Acrylic Sheet	Lowe's	78778	$34.88
2	Plastic Dip Spray Red	Lowe's	150123	$17.08
1	Petmate Replendish	Amazon	24539	$9.26
1	Plastic Diy Robot Gear Kit	Amazon	B06W9FSTCP	$10.96
1	Taste of Wild Grain	Amazon	1998	$10.69
1	Zener Diode	Amazon	B07BTKVRG8	$7.88
1	Push buttons	Amazon	B07F8GBWGG	$8.89
1	Voltage Regulator	Amazon	B07BDFMQF6	$6.99
1	Arduino Mega 2560 R3	Amazon	B01H4ZLZLQ	$14.99
1	Servo Motor	Amazon	B07KR24YX8	$12.99
1	Voice Recognition Module	Amazon	800-001-0014	$35
1	LCD 20x4 I2C	Amazon	B07QLRD3TM	$7.98
1	Ultrasonic Sensor	Amazon	B004U8TOE6	$4.99
1	Kastar AC Adapter	Amazon	LSE0111BB1275	$10.99
2	Printed Circuit Board	IUPUI	DAK	$0
	Total Price			$241.60

Pet Feeder with Pet Identification

The remaining section of this document is the fully operable code to the feeder machine. This code is confidential and is to only be used for educational purposes only. The code was copied directly from Arduino, so the format won't look correct, but it is.

```
#include <EEPROM.h>

#include <DS3232RTC.h>

#include <avr/sleep.h>
#include <NewPing.h>
#include <LCD.h>
#include <Wire.h>
#include <Servo.h>
#include <LiquidCrystal_I2C.h>

#include <LowPower.h>
#include <PinChangeInterrupt.h>
#include <RTClib.h>

#define rtcAlarmPin 10 // External interrupt on pin D3

 // time conversion
/*#define ms_per_hour  3600000
#define ms_per_min    60000
#define ms_per_sec    1000
*/

RTC_DS3231 rtc;
volatile bool alarmNow = false;

 int addr1 = 0;
 int addr2 = 1;
 int addr3 = 2;
 int addr4 = 3;

#define lcdAddress 0x27
#define lcdColumns 20
#define lcdRows 4
#define TRIGGER_PIN  A8
#define ECHO_PIN     A9
#define MAX_DISTANCE 200
```

Pet Feeder with Pet Identification

```
#define PB_FoodSize 6
#define PB_TimeSpan 7
#define PB_Test 8

#define interruptPin 2 // Pin to wake up arduino

// Assigned digit al pins

const byte GreenLED = 9;
const byte YellowLED = 5;
const byte RedLED = 11;
const byte ServoPin = 12;

volatile boolean sw1 = false;
uint8_t sw1ButtonState = 0;
uint8_t lastsw1ButtonState = 0;
int ButtonPress ;

volatile boolean sw2 = false;

int Time_interval;

volatile boolean Feeder_Open = false;
volatile boolean Feeder_Close = false;
volatile boolean Sleep_Switch = false;
volatile boolean Sleep_OK = false;
volatile boolean Wake_UP_OK = false;

volatile boolean sw3 = false;

int Test_Count = 0;

// variables declared
int RAWdistance = 0;
int distance = 0;
int ServoPosition = 0;

byte incomingData = 0;

unsigned long previousTime = 0;
```

Pet Feeder with Pet Identification

```
const long Interval =180000;

unsigned long previousTime2 = 0;
const long Interval2 = 240000;

unsigned long previousTime3 = 0;
const long Interval3 = 300000;

  unsigned long currentTime = 0;
  unsigned long currentTime2 = 0;
  unsigned long currentTime3 = 0;
  unsigned long elapse1 = 0;
  unsigned long elapse2 = 0;
  unsigned long elapse3 = 0;
 unsigned long time_counter  = 0;

 bool timeReceived = false;
unsigned long lastUpdate=0, lastRequest=0;

  unsigned long Delay_Sleep = 0;
  unsigned long Delay_Sleep_Interval = 30000;
  unsigned long Delay_WakeUp =0;
  unsigned long Delay_To_Go_Sleep = 0;
  unsigned long Now_Wake_Up = 0;


// Set the pins on the I2C chip used for LCD connections:
// addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(lcdAddress, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);

//LiquidCrystal_I2C lcd(0x3F,20,4);

NewPing sonar(TRIGGER_PIN, ECHO_PIN, MAX_DISTANCE);


Servo MyServo;  // create servo object to control a servo


void setup()
{
  //    lcd.init();
```

```
Wire.begin();

// set up time only once


// The following lines can be uncommented to set the date and time

lcd.begin(lcdColumns, lcdRows);
lcd.backlight();
lcd.clear();
Serial.begin(9600);
MyServo.attach(12);  // attaches the servo pin 10 to the servo object
MyServo.write (0); // servo position 0
pinMode(13, OUTPUT);
 CloseReservoir();
//pinMode(PB_FoodSize, INPUT_PULLUP);
pinMode(PB_FoodSize, INPUT_PULLUP);
pinMode(PB_TimeSpan, INPUT_PULLUP);
pinMode(PB_Test, INPUT_PULLUP);
pinMode(interruptPin, INPUT_PULLUP);

pinMode(GreenLED, OUTPUT);
pinMode(YellowLED, OUTPUT);
pinMode(RedLED, OUTPUT);
lcd.begin(lcdColumns, lcdRows);

delay(2000);
Serial.write(0xAA);
Serial.write(0x37);
delay(1000);
Serial.write(0xAA);
Serial.write(0x21);

 // the function to get the time from the RTC
setSyncProvider(RTC.get);

//Serial.begin(115200);
pinMode(rtcAlarmPin, INPUT_PULLUP); // Set interrupt pin
attachPCINT(digitalPinToPCINT(rtcAlarmPin), WakeUp, FALLING);
// initialize the alarms to known values, clear the alarm flags, clear the alarm interrupt flags
RTC.setAlarm(ALM1_MATCH_DATE, 0, 0, 0, 1);
```

```
  RTC.setAlarm(ALM2_MATCH_DATE, 0, 0, 0, 1);
  RTC.alarm(ALARM_1);
  RTC.alarm(ALARM_2);
  RTC.alarmInterrupt(ALARM_1, false);
  RTC.alarmInterrupt(ALARM_2, false);
  RTC.squareWave(SQWAVE_NONE);

    // set the alarm

 lcd.clear();
  DateTime now = rtc.now();
  printDateTime(now);
  DateTime future (now + TimeSpan(0, 0, 0, 30)); //date, hour, minute, Second
   printDateTime(future);

  RTC.setAlarm(ALM1_MATCH_DATE, future.second(), future.minute(), future.hour(),
future.day());
  // clear the alarm flag
  RTC.alarm(ALARM_1);
  RTC.alarmInterrupt(ALARM_1, true); // Enable alarm 1 interrupt A1IE
 lcd.clear();


    ButtonPress = EEPROM.read(addr1);  // read foode size
    Time_interval = EEPROM.read(addr2); // read time interval
    Test_Count =  EEPROM.read(addr3); // read test counter

digitalWrite(13, HIGH);
digitalWrite(RedLED, HIGH);


if (ButtonPress == 0 && Time_interval == 0 )
 { lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("select food size");
  lcd.setCursor(0, 1);
  lcd.print("and time interval");
   delay(1500);
 }

}
```

```
void receiveTime(unsigned long controllerTime) {
  // Ok, set incoming time
  //Serial.print("Time value received: ");
 // Serial.println(controllerTime);
  RTC.set(controllerTime); // this sets the RTC to the time from controller - which we do want
periodically
  timeReceived = true;

  lcd.clear();
}


void loop()
{


 unsigned long now = millis();

  // If no time has been received yet, request it every 10 second from controller
  // When time has been received, request update every hour
  if ((!timeReceived && (now-lastRequest) > (10UL*1000UL))
    || (timeReceived && (now-lastRequest) > (60UL*1000UL*60UL))) {
    // Request time from controller.
    //  Serial.println("requesting time");
   //    requestTime();
    lastRequest = now;
  }

  // Update display every second
  if (now-lastUpdate > 1000) {
    updateDisplay();
    lastUpdate = now;
  }


  Get_Food_size();
  Time_Interval();
  Test_Mode();

  updateDisplay();
```

```
 tmElements_t tm;
  RTC.read(tm);

//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Small size,
inteval 3h (2,2)
  if (ButtonPress == 2 && Time_interval == 2)

  {
     Feeder_Open = false;
    Feeder_Close = false;

   lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("small size ");
     lcd.print("3h");
    delay(1000);

          Get_Distance();

         Time_Interval_1();

        elapse1 = (currentTime - previousTime);
           lcd.setCursor(0, 2);
           lcd.print(elapse1);
           lcd.setCursor(5, 2);
         lcd.print("min");
            delay(1000);

           if (distance  <= 25)    // 25 cm
                {

                         Feeder_Open = false;
                         Feeder_Close = false;
                        if (Serial.available())
                              {

                                incomingData = Serial.read();

                                Feeder_Open = false;
```

Pet Feeder with Pet Identification

```
Feeder_Close = false;


 // calculate the current time base on the last food dispense time
// currentTime = currentTime+ previousTime;



  if((currentTime - previousTime)>= 3)

 {
 lcd.clear();
  lcd.setCursor(0, 2);
  lcd.print("Interval OK");
    delay(1000);


     Feeder_Open = false;
     Feeder_Close = false;



  switch (incomingData)

 {
  case 0x11:   // command 1,

   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("This is my");
   lcd.setCursor(0, 1);
   lcd.print("cat's voice");

    delay(2000);

     lcd.clear();


    OpenReservoir();
     CloseReservoir();
```

Pet Feeder with Pet Identification

```
  currentTime = 0;

/* lcd.setCursor(0, 3);
  lcd.print(previousTime);
  lcd.setCursor(5, 2);
  lcd.print("min");
  delay(1000);
   */

   Delay_To_Go_Sleep = millis();

 break;

/*  case 0x12: //command 2 Turn ON Yellow LED

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my's");
  lcd.setCursor(0, 1);
  lcd.print("cat voice");
  delay(1000);


  OpenReservoir();
  delay(3000);
  CloseReservoir();

   break;

  case 0x13: //command 3 Turn ON Red LED

  break;

   case 0x14: //command 4 open dish

  break;

 case 0x15: //command 5 close Dish
```

```
                    break;*/

                   }

                  }

                 }

          }

           if (Delay_To_Go_Sleep >= 30000  )
          {

             Now_Wake_Up =  millis();
              Go_TO_Sleep();
             delay(120000);

            }
         if ( Now_Wake_Up >= 120000 )
         {
             setRTCagain();

             Delay_To_Go_Sleep = 0;
            Now_Wake_Up = 0;
         }
 }
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Small size,
interval 4h (2,3)

  if (ButtonPress == 2 && Time_interval == 3)
  {

    lcd.clear();
```

```
lcd.setCursor(0, 0);
lcd.print("small size");
lcd.print(" ");
lcd.print("4h ");
delay(1000);


        Get_Distance();

         Time_Interval_2();
          Feeder_Open = false;
           Feeder_Close = false;


    lcd.setCursor(0, 2);
    lcd.print( currentTime2 - previousTime2);
    lcd.setCursor(5, 2);
    lcd.print("min");
     delay(1000);


     if (distance  <= 25)
         {

                      Feeder_Open = false;
                      Feeder_Close = false;
              if (Serial.available())
                   {

                    incomingData = Serial.read();
                      Feeder_Open = false;
                      Feeder_Close = false;

                     if(currentTime2 - previousTime2 >= 4)

                     {

                        lcd.setCursor(0, 2);
                       lcd.print("Interval OK");
                        delay(1000);
```

Pet Feeder with Pet Identification

```
  Feeder_Open = false;
  Feeder_Close = false;



switch (incomingData)

{
 case 0x11:   // command 1,

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my");
  lcd.setCursor(0, 1);
  lcd.print("cat's voice");
   delay(2000);
    lcd.clear();


  OpenReservoir();
  CloseReservoir();

  lcd.setCursor(0, 3);
  lcd.print(previousTime2);
  lcd.setCursor(5, 2);
  lcd.print("min");
  delay(1000);

   Delay_To_Go_Sleep = millis();

 break;

/*  case 0x12: //command 2 Turn ON Yellow LED

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my's");
  lcd.setCursor(0, 1);
  lcd.print("cat voice");
  delay(1000);
```

Pet Feeder with Pet Identification

```
                OpenReservoir();
                delay(3000);
                CloseReservoir();

                 break;

                case 0x13: //command 3 Turn ON Red LED

                break;

                 case 0x14: //command 4 open dish

                break;

               case 0x15: //command 5 close Dish

               break;*/


              }


            }


          }


        }

        if (Delay_To_Go_Sleep >= 30000  )
{


   Now_Wake_Up =  millis();
```

Pet Feeder with Pet Identification

```
            Go_TO_Sleep();
           delay(180000);



          }
      if ( Now_Wake_Up >= 180000 )
      {
           setRTCagain();

           Delay_To_Go_Sleep = 0;
          Now_Wake_Up = 0;
      }

    }


//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Small size,
inteval 5h (2,4)

 if (ButtonPress == 2 && Time_interval == 4)
 {

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("small size ");
  lcd.print("5h");
  delay(1000);


     Get_Distance();

     Time_Interval_3();
     Feeder_Open = false;
      Feeder_Close = false;
        elapse3 = currentTime3 - previousTime3;

     lcd.setCursor(0, 2);
     lcd.print(elapse3);
     lcd.setCursor(5, 2);
```

Pet Feeder with Pet Identification

```
lcd.print("min");
 delay(1000);


 if (distance  <= 25)
     {

                    Feeder_Open = false;
                    Feeder_Close = false;
            if (Serial.available())
                 {
                  incomingData = Serial.read();
                    Feeder_Open = false;
                    Feeder_Close = false;



                  if(currentTime3 - previousTime3 > 5)

                 {
                    lcd.setCursor(0, 2);
                   lcd.print("Interval OK");
                   delay(1000);

                    Feeder_Open = false;
                    Feeder_Close = false;


                  switch (incomingData)

                  {
                   case 0x11:   // command 1,


                    lcd.clear();
                    lcd.setCursor(0, 0);
                    lcd.print("This is my");
                    lcd.setCursor(0, 1);
                    lcd.print("cat's voice");

                     delay(2000);
```

Pet Feeder with Pet Identification

```
 lcd.clear();


 OpenReservoir();

 CloseReservoir();

 lcd.setCursor(0, 3);
 lcd.print(previousTime3);
 lcd.setCursor(5, 2);
 lcd.print("min");
 delay(1000);




Delay_To_Go_Sleep = millis();



break;

/*  case 0x12: //command 2 Turn ON Yellow LED

 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("This is my's");
 lcd.setCursor(0, 1);
 lcd.print("cat voice");
 delay(1000);


 OpenReservoir();
 delay(3000);
 CloseReservoir();

 break;

 case 0x13: //command 3 Turn ON Red LED
```

Pet Feeder with Pet Identification

```
                    break;
                     case 0x14: //command 4 open dish
                    break;
                   case 0x15: //command 5 close Dish
                   break;*/


                  }


                 }


               }


            }


          if (Delay_To_Go_Sleep >= 30000  )
    {


       Now_Wake_Up =  millis();
         Go_TO_Sleep();
       delay(240000);



     }
if ( Now_Wake_Up > 2400000 )
{
      setRTCagain();
```

```
                Delay_To_Go_Sleep = 0;
               Now_Wake_Up = 0;
          }

                 }


//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Medium size,
inteval 3h (3,2)

  if (ButtonPress == 3 && Time_interval == 2)
  {

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Medium size ");
    lcd.print("3h ");
    delay(1000);



        Get_Distance();

      Time_Interval_1();

             Feeder_Open = false;
             Feeder_Close = false;

         elapse1 = currentTime - previousTime;
        lcd.setCursor(0, 2);
        lcd.print(elapse1);
        lcd.setCursor(5, 2);
        lcd.print("min");
         delay(1000);


         if (distance  <= 25)
              {
```

Pet Feeder with Pet Identification

```
            Feeder_Open = false;
            Feeder_Close = false;
if (Serial.available())
     {

       incomingData = Serial.read();
            Feeder_Open = false;
            Feeder_Close = false;



         if(currentTime - previousTime >= 3)

        {


          lcd.setCursor(0, 2);
          lcd.print("Interval OK");
          delay(1000);

            Feeder_Open = false;
          Feeder_Close = false;



        switch (incomingData)

        {
         case 0x11:   // command 1,

          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("This is my");
          lcd.setCursor(0, 1);
          lcd.print("cat's voice");

           delay(2000);
            lcd.clear();


          OpenReservoir2();
```

```
  CloseReservoir2();
   ;


  lcd.setCursor(0, 3);
  lcd.print(previousTime);
  lcd.setCursor(5, 2);
  lcd.print("min");
  delay(2000);

   Delay_To_Go_Sleep = millis();

 break;

/*  case 0x12: //command 2 Turn ON Yellow LED

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my's");
  lcd.setCursor(0, 1);
  lcd.print("cat voice");
  delay(1000);


  OpenReservoir();
  delay(3000);
  CloseReservoir();

   break;

  case 0x13: //command 3 Turn ON Red LED

  break;

   case 0x14: //command 4 open dish

  break;

 case 0x15: //command 5 close Dish
```

Pet Feeder with Pet Identification

```
                    break;*/


                   }


                  }


                 }


              }

             if (Delay_To_Go_Sleep >= 30000  )
        {


           Now_Wake_Up =  millis();
            Go_TO_Sleep();
           delay(120000);



         }
       if ( Now_Wake_Up >= 120000 )
       {
            setRTCagain();

            Delay_To_Go_Sleep = 0;
           Now_Wake_Up = 0;
       }




}
```

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Medium size,
inteval 4h (3,3)

  if (ButtonPress == 3 && Time_interval == 3)
  {

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Medium size ");
       lcd.print("4h");
    delay(1000);


          Get_Distance();

         Time_Interval_2();

         Feeder_Open = false;
         Feeder_Close = false;

         elapse2 = currentTime2 - previousTime2;
          lcd.setCursor(0, 2);
          lcd.print(elapse2);
          lcd.setCursor(5, 2);
          lcd.print("min");
           delay(1000);


           if (distance  <= 25)
                {

                                Feeder_Open = false;
                                Feeder_Close = false;
                      if (Serial.available())
                            {
                              incomingData = Serial.read();
                                Feeder_Open = false;
                                Feeder_Close = false;
```

Pet Feeder with Pet Identification

```
if(currentTime2 - previousTime2 >= 4)

{


  lcd.setCursor(0, 2);
  lcd.print("Interval OK");
  delay(1000);

    Feeder_Open = false;
   Feeder_Close = false;


switch (incomingData)

{
 case 0x11:   // command 1,

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my");
  lcd.setCursor(0, 1);
  lcd.print("cat's voice");

   delay(2000);
    lcd.clear();

  OpenReservoir2();

   CloseReservoir2();
    ;

   lcd.setCursor(0, 3);
   lcd.print(previousTime2);
   lcd.setCursor(5, 2);
   lcd.print("min");
   delay(2000);
```

Pet Feeder with Pet Identification

```
    Delay_To_Go_Sleep = millis();

  break;

 /*  case 0x12: //command 2 Turn ON Yellow LED

   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("This is my's");
   lcd.setCursor(0, 1);
   lcd.print("cat voice");
   delay(1000);


   OpenReservoir();
   delay(3000);
   CloseReservoir();

    break;

   case 0x13: //command 3 Turn ON Red LED

   break;

    case 0x14: //command 4 open dish

   break;

  case 0x15: //command 5 close Dish

  break;*/


 }


}
```

```
                }


            }


                if (Delay_To_Go_Sleep >= 30000  )
            {



              Now_Wake_Up =  millis();
               Go_TO_Sleep();
               delay(180000);



              }
          if ( Now_Wake_Up >= 180000 )
          {
               setRTCagain();

               Delay_To_Go_Sleep = 0;
              Now_Wake_Up = 0;
          }

     }

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Medium size, inteval 5h (3,4)

  if (ButtonPress == 3 && Time_interval == 4)
  {

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print(" Midium size ");
     lcd.print("5h ");
    delay(1000);
```

Pet Feeder with Pet Identification

```
 Get_Distance();
Time_Interval_3();
 Feeder_Open = false;
 Feeder_Close = false;

elapse3 = currentTime3 - previousTime3;
 lcd.setCursor(0, 2);
 lcd.print(elapse3);
 lcd.setCursor(5, 2);
 lcd.print("min");
 delay(1000);


 if (distance  <= 25)
     {

                    Feeder_Open = false;
                    Feeder_Close = false;
            if (Serial.available())
                {
                 incomingData = Serial.read();
                    Feeder_Open = false;
                    Feeder_Close = false;



                  if(currentTime3 - previousTime3 >= 5)

                 {

                   lcd.setCursor(0, 2);
                   lcd.print("Interval OK");
                   delay(1000);



                    Feeder_Open = false;
                    Feeder_Close = false;
```

```
switch (incomingData)

{
 case 0x11:   // command 1,


  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my");
  lcd.setCursor(0, 1);
  lcd.print("cat's voice");

   delay(2000);
    lcd.clear();


  OpenReservoir2();

   CloseReservoir2();


   lcd.setCursor(0, 3);
   lcd.print(previousTime3);
   lcd.setCursor(5, 2);
   lcd.print("min");
   delay(2000);

 Delay_To_Go_Sleep = millis();



 break;

/*  case 0x12: //command 2 Turn ON Yellow LED

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my's");
  lcd.setCursor(0, 1);
```

Pet Feeder with Pet Identification

```
                lcd.print("cat voice");
                delay(1000);


                OpenReservoir();
                delay(3000);
                CloseReservoir();

                 break;

                case 0x13: //command 3 Turn ON Red LED

                break;

                 case 0x14: //command 4 open dish

                break;

               case 0x15: //command 5 close Dish

               break;*/



              }


            }


          }


       }
     if (Delay_To_Go_Sleep >=30000  )
{


  Now_Wake_Up =  millis();
```

```
            Go_TO_Sleep();
           delay(240000);



         }
      if ( Now_Wake_Up >= 240000 )
      {
           setRTCagain();

           Delay_To_Go_Sleep = 0;
          Now_Wake_Up = 0;
      }
     }


//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Large size,
inteval 3h (4,2)

 if (ButtonPress == 4 && Time_interval == 2)
 {

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Large size ");
    lcd.print("3h");
  delay(1000);


     Get_Distance();

     Time_Interval_1();

         Feeder_Open = false;
         Feeder_Close = false;


     elapse1 = currentTime - previousTime;

      lcd.setCursor(0, 2);
```

Pet Feeder with Pet Identification

```
lcd.print(elapse1);
lcd.setCursor(5, 2);
lcd.print("min");
 delay(1000);


 if (distance  <= 25)
     {

          Feeder_Open = false;
           Feeder_Close = false;

            if (Serial.available())
                {
                 incomingData = Serial.read();
                     Feeder_Open = false;
                     Feeder_Close = false;

                  if(currentTime - previousTime >= 3)

                  {

                    lcd.setCursor(0, 2);
                    lcd.print("Interval OK");
                    delay(1000);


                      Feeder_Open = false;
                     Feeder_Close = false;



                  switch (incomingData)

                  {
                   case 0x11:   // command 1,

                    lcd.clear();
                    lcd.setCursor(0, 0);
                    lcd.print("This is my");
                    lcd.setCursor(0, 1);
```

```
 lcd.print("cat's voice");

  delay(2000);
   lcd.clear();


    OpenReservoir3();

   CloseReservoir3();


   lcd.setCursor(0, 3);
   lcd.print(previousTime);
   lcd.setCursor(5, 2);
   lcd.print("min");
   delay(1000);

  Delay_To_Go_Sleep = millis();

  break;

 /*  case 0x12: //command 2 Turn ON Yellow LED

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my's");
  lcd.setCursor(0, 1);
  lcd.print("cat voice");
  delay(1000);


  OpenReservoir();
  delay(3000);
  CloseReservoir();

   break;

  case 0x13: //command 3 Turn ON Red LED

  break;
```

Pet Feeder with Pet Identification

```
                    case 0x14: //command 4 open dish

                   break;

                  case 0x15: //command 5 close Dish

                  break;*/


                 }


               }


             }


          }

           if (Delay_To_Go_Sleep >= 30000  )
   {

      Now_Wake_Up =  millis();
        Go_TO_Sleep();
       delay(120000);

     }


 if ( Now_Wake_Up >= 120000 )
 {
       setRTCagain();

       Delay_To_Go_Sleep = 0;
      Now_Wake_Up = 0;
 }

}
```

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Large size,
inteval 4h(4,3)
  if (ButtonPress == 4 && Time_interval == 3)
  {

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Large size");
    lcd.print(" ");
    lcd.print("4h ");
    delay(1000);


         Get_Distance();
        Time_Interval_2();

        Feeder_Open = false;
        Feeder_Close = false;

        elapse2 = currentTime2 - previousTime2;
         lcd.setCursor(0, 2);
         lcd.print(elapse2);
         lcd.setCursor(5, 2);
         lcd.print("min");
          delay(1000);


          if (distance  <= 25)
               {

                              Feeder_Open = false;
                              Feeder_Close = false;
                   if (Serial.available())
                         {
                           incomingData = Serial.read();
                              Feeder_Open = false;
                              Feeder_Close = false;
```

Pet Feeder with Pet Identification

```
if(currentTime2 - previousTime2 >= 4)

{

   lcd.setCursor(0, 2);
  lcd.print("Interval OK");
  delay(1000);


    Feeder_Open = false;
   Feeder_Close = false;



switch (incomingData)

{
 case 0x11:   // command 1,

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my");
  lcd.setCursor(0, 1);
  lcd.print("cat's voice");

   delay(2000);
    lcd.clear();

  OpenReservoir3();
  CloseReservoir3();

   lcd.setCursor(0, 3);
   lcd.print(previousTime2);
   lcd.setCursor(5, 2);
   lcd.print("min");
   delay(1000);
```

```
    Delay_To_Go_Sleep = millis();

  break;

 /*  case 0x12: //command 2 Turn ON Yellow LED

   lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("This is my's");
   lcd.setCursor(0, 1);
   lcd.print("cat voice");
   delay(1000);


   OpenReservoir();
   delay(3000);
   CloseReservoir();

    break;

   case 0x13: //command 3 Turn ON Red LED

   break;

    case 0x14: //command 4 open dish

   break;

  case 0x15: //command 5 close Dish

  break;*/


 }


}
```

Pet Feeder with Pet Identification

```
                    }


                }
               if (Delay_To_Go_Sleep >= 30000  )
          {



             Now_Wake_Up =  millis();
               Go_TO_Sleep();
              delay(180000);



            }
         if ( Now_Wake_Up > 180000 )
         {
              setRTCagain();

              Delay_To_Go_Sleep = 0;
             Now_Wake_Up = 0;
         }
       }


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++ Large size,
inteval 5h(4,4)

  if (ButtonPress == 4 && Time_interval == 4)
  {

    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Large size ");
    lcd.print("5h");
    delay(1000);
```

Pet Feeder with Pet Identification

```
 Get_Distance();

Time_Interval_3();

Feeder_Open = false;
 Feeder_Close = false;

elapse3 = currentTime3 - previousTime3;
 lcd.setCursor(0, 2);
 lcd.print(elapse3);
 lcd.setCursor(5, 2);
 lcd.print("min");
  delay(1000);


  if (distance  <= 25)
      {

                    Feeder_Open = false;
                    Feeder_Close = false;
            if (Serial.available())
                {
                  incomingData = Serial.read();
                    Feeder_Open = false;
                    Feeder_Close = false;



                   if(currentTime3 - previousTime3 >= 5)

                   {

                      lcd.setCursor(0, 2);
                     lcd.print("Interval OK");
                     delay(1000);


                      Feeder_Open = false;
                     Feeder_Close = false;
```

```
switch (incomingData)

{
 case 0x11:   // command 1,

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my");
  lcd.setCursor(0, 1);
  lcd.print("cat's voice");

   delay(2000);
    lcd.clear();


  OpenReservoir3();

   CloseReservoir3();

   lcd.setCursor(0, 3);
   lcd.print(previousTime3);
   lcd.setCursor(5, 2);
   lcd.print("min");
   delay(1000);


 Delay_To_Go_Sleep = millis();


 break;

/*  case 0x12: //command 2 Turn ON Yellow LED

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my's");
  lcd.setCursor(0, 1);
  lcd.print("cat voice");
```

Pet Feeder with Pet Identification

```
                delay(1000);


                OpenReservoir();
                delay(3000);
                CloseReservoir();

                 break;

                case 0x13: //command 3 Turn ON Red LED

                break;

                 case 0x14: //command 4 open dish

                break;

               case 0x15: //command 5 close Dish

               break;*/


              }


            }


          }


        }

        if (Delay_To_Go_Sleep >=30000  )
{
```

```
            Now_Wake_Up =  millis();
              Go_TO_Sleep();
             delay(240000);



          }
        if ( Now_Wake_Up >= 240000 )
        {
              setRTCagain();

              Delay_To_Go_Sleep = 0;
             Now_Wake_Up = 0;
        }

      }

 //
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++ Test mode function

        if ( Test_Count == 2)

          {
           sw3 = true;

         //  ButtonPress = 0;
           Time_interval = 0;
           lcd.backlight(); // turn on backlight.
            lcd.clear();
           lcd.setCursor(0, 0);
           lcd.print("Test Mode Activate");
           lcd.setCursor(0, 1);
           lcd.print("LCD Screen OK");
           delay(2000);



            delay(1000);
```

```
        lcd.clear();
       }
       else
       {
        sw3 = false;
       }


    if ( Test_Count == 3)

       {
        sw3 = true;

        ButtonPress = 0;
        Time_interval = 0;

        lcd.clear();
       lcd.setCursor(0, 0);
        lcd.print("Test Mode Activate");

       Get_Distance();

        lcd.setCursor(0, 2);
        lcd.print("Sensor OK");

         delay(2000);
         lcd.clear();
       }

  if ( Test_Count == 4)

       {
        sw3 = true;

        ButtonPress = 0;
        Time_interval = 0;

         lcd.clear();
       lcd.setCursor(0, 0);
       lcd.print("Test Mode Activate");
```

```
 delay(1000);
 lcd.clear();
 OpenReservoir();
 CloseReservoir();

 lcd.setCursor(0, 2);
lcd.print("Servo OK");
delay(2000);
}


if ( Test_Count == 5)

{
sw3 = true;

ButtonPress = 0;
Time_interval = 0;
 lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Test Mode Activate");
 lcd.setCursor(0, 1);
lcd.print("Play cat voice ");
 delay(2000);
 lcd.clear();



    if (Serial.available())
          {


          incomingData = Serial.read();



          switch (incomingData)

          {
           case 0x11:   // command 1,
```

Pet Feeder with Pet Identification

```
                    lcd.clear();
                     lcd.setCursor(0, 0);
                     lcd.print("This is my");
                     lcd.setCursor(0, 1);
                     lcd.print("cat's voice");


                     lcd.setCursor(0, 2);
                     lcd.print("voice Recogn OK");
                      delay(2000);
                      break;
                    }
                 }
            }

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Default feeding

      if (ButtonPress == 1 && Time_interval == 1)

  {

   Sleep_Switch = false;

  Feeder_Open = false;
   Feeder_Close = false;
  lcd.clear();
   lcd.setCursor(0, 0);
   lcd.print("default size / Hr");
     delay(1000);

        Get_Distance();

       Time_Interval_1();

         lcd.setCursor(0, 2);
         lcd.print(currentTime - previousTime);
```

Pet Feeder with Pet Identification

```
lcd.setCursor(8, 2);
 lcd.print("min");
 delay(1000);

if (distance  <= 25)
    {

          Feeder_Open = false;
          Feeder_Close = false;
         if (Serial.available())
             {
                 Feeder_Open = false;
                 Feeder_Close = false;

           incomingData = Serial.read();

           // calculate the current time base on the last food dispense time
          // currentTime = currentTime+ previousTime;



            if(currentTime - previousTime >= Interval)

           {

             lcd.setCursor(0, 2);
             lcd.print("Interval OK");
               delay(1000);

                Feeder_Open = true;
                Feeder_Close = true;

              lcd.clear();


               OpenReservoir();
               delay(3000);
               CloseReservoir();

                 Delay_To_Go_Sleep = millis();
```

Pet Feeder with Pet Identification

```
/*switch (incomingData)

{
 case 0x11:   // command 1,

 lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my");
  lcd.setCursor(0, 1);
  lcd.print("cat's voice");

   delay(1000);

    lcd.clear();


   OpenReservoir();
   delay(3000);
   CloseReservoir();

   Delay_To_Go_Sleep = millis();
  Now_Wake_Up =  millis();
// previousTime = currentTime;
   lcd.setCursor(0, 3);
   lcd.print(previousTime);
   lcd.setCursor(7, 2);
   lcd.print("ms");
   delay(2000);


   Feeder_Open = true;
   Feeder_Close = true;

 break;

/*  case 0x12: //command 2 Turn ON Yellow LED

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("This is my's");
  lcd.setCursor(0, 1);
```

Pet Feeder with Pet Identification

```
                lcd.print("cat voice");
                delay(1000);


                OpenReservoir();
                delay(3000);
                CloseReservoir();

                 break;

                case 0x13: //command 3 Turn ON Red LED

                break;

                 case 0x14: //command 4 open dish

                break;

               case 0x15: //command 5 close Dish

               break;*/


              }


            }


          }


        }


  if (Delay_To_Go_Sleep >= 30000  )
{
```

Pet Feeder with Pet Identification

```
            Go_TO_Sleep();
           delay(12000);
      }
        if ( Now_Wake_Up >= 120000 )
    {

       setRTCagain();
    Delay_To_Go_Sleep = 0;
     Now_Wake_Up = 0;
    }


// Reset EEPROM Memory
if (ButtonPress == 1 && Time_interval == 1&& Test_Count == 1)
{

 EEPROM.write(0, 0);
 EEPROM.write(1, 0);
 EEPROM.write(2, 0);

 lcd.clear();
 lcd.setCursor(0, 0);
 lcd.print("EEPROM Cleared");
 lcd.setCursor(0, 1);
 lcd.print("Cycle Power Feeder");
 delay(2000);

ButtonPress = 0;
Time_interval  = 0;

}



}
```

Pet Feeder with Pet Identification

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ Functions

void Get_Food_size()
{
  if (digitalRead(PB_FoodSize) == LOW)
  {
    sw1 = true;
    ButtonPress = ButtonPress + 1;

  EEPROM.write(addr1, ButtonPress);
  ButtonPress =  EEPROM.read(addr1);
  if (EEPROM.read(addr1) != ButtonPress)
     {
   // value1 = ButtonPress;
     EEPROM.write(addr1, ButtonPress);
     }

    lcd.clear();
    lcd.setCursor(0, 3);
    lcd.print(ButtonPress);
    lcd.print(" ");
    delay(1000);
    lcd.clear();
  }
  else
  {
    sw1 = false;
  }


  if (ButtonPress == 5)
  {
    ButtonPress = 1;
    lcd.clear();
    lcd.setCursor(5, 2);
    lcd.print("size reset to 1");
     delay(1500);
    lcd.clear();
  }
```

```
}


void Time_Interval()
{

  if (digitalRead(PB_TimeSpan) == LOW)
  {
    sw2 = true;
    Time_interval = Time_interval + 1;

   EEPROM.write(addr2, Time_interval);

   Time_interval =  EEPROM.read(addr2);
  if (EEPROM.read(addr2) != Time_interval)
     {

     EEPROM.write(addr2, Time_interval);
     }



    lcd.clear();
    lcd.setCursor(0, 3);
    lcd.print(Time_interval);
    delay(1000);
    lcd.clear();
  }
  else
  {
    sw2 = false;
  }


  if (Time_interval == 5)
  {
    Time_interval = 1;
    lcd.clear();
    lcd.setCursor(3, 3);
    lcd.print("Interval Reset to 1");
```

```
   delay(1500);
   lcd.clear();
  }
}

void Test_Mode()
{

  if (digitalRead(PB_Test) == LOW)
  {
    sw3 = true;
    Test_Count = Test_Count + 1;




      EEPROM.write(addr3, Test_Count);

   Test_Count =  EEPROM.read(addr3);
  if (EEPROM.read(addr3) != Test_Count)
     {

     EEPROM.write(addr3, Test_Count);
     }


   lcd.clear();
   lcd.setCursor(0, 3);
   lcd.print(Test_Count);
   delay(1000);
   lcd.clear();
  }
  else
  {
   sw3 = false;
  }


  if (Test_Count == 6 )
  {
   Test_Count = 1;
```

```
   lcd.clear();
   lcd.setCursor(0, 2);
   lcd.print("Test_Count");
   lcd.setCursor(0, 3);
   lcd.print("Reset to 1");
   delay(1500);
   lcd.clear();
  }
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++1
void CloseReservoir()
{
  Sleep_OK = false;
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print(" food dispensed ");
  // ServoPosition = 180; // set degree to 180
  digitalWrite(GreenLED, LOW);
  digitalWrite(YellowLED, HIGH); // turn  on Yellow led while  gate is closing
  for (ServoPosition == 75; ServoPosition >= 0; ServoPosition -= 1)
  { // goes from 180 degrees to 0 degrees
    MyServo.write(ServoPosition);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position

    Feeder_Close = true;
    previousTime =  currentTime;
    previousTime2 =  currentTime2;
    previousTime3 =  currentTime3;

  // reset timers each day
     if ( currentTime == 1440)
     {
      currentTime = 0;
     }


     if ( currentTime2 == 1440)
     {
      currentTime2 = 0;
```

```
    }
    if ( currentTime3 == 1440)
    {
     currentTime3 = 0;
    }
  }
  digitalWrite(YellowLED, LOW);
  digitalWrite(RedLED, HIGH);// turn on led red when gate is closed
  delay(3000);

   MyServo.detach();
}



void OpenReservoir()
{

  MyServo.attach(12);
  delay(100);
  Sleep_OK = false;
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print("feeder is opening");
  delay(1000);
  lcd.clear();

  digitalWrite(RedLED, LOW);
  digitalWrite(YellowLED, HIGH); // turn  on Yellow led while  gate is opening

  // sets the servo position according to the scaled value
  for (ServoPosition = 0; ServoPosition <= 75; ServoPosition += 1)
  { // goes from 0 degrees to 180 degrees steps of 1 degree

    MyServo.write(ServoPosition);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15ms for the servo to reach the position

    Feeder_Open = true;
```

```
  }

  digitalWrite(YellowLED, LOW);
  digitalWrite(GreenLED, HIGH); // turn on green led when gate opened
  lcd.setCursor(0, 0);
  lcd.print("Feeder is Opened");
  delay(3000);
  lcd.clear();

}


//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++2

void CloseReservoir2()
{
  Sleep_OK = false;
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print(" food dispensed ");
  // ServoPosition = 180; // set degree to 180
  digitalWrite(GreenLED, LOW);
  digitalWrite(YellowLED, HIGH); // turn  on Yellow led while  gate is closing
  for (ServoPosition == 75; ServoPosition >= 0; ServoPosition -= 1)
  { // goes from 180 degrees to 0 degrees
    MyServo.write(ServoPosition);              // tell servo to go to position in variable 'pos'
    delay(20);                      // waits 15ms for the servo to reach the position

    Feeder_Close = true;
    previousTime =  currentTime;
    previousTime2 =  currentTime2;
    previousTime3 =  currentTime3;

  // reset timers each day
     if ( currentTime == 1440)
     {
      currentTime = 0;
     }
```

```
    if ( currentTime2 == 1440)
    {
     currentTime2 = 0;
    }
    if ( currentTime3 == 1440)
    {
     currentTime3 = 0;
    }
  }
  digitalWrite(YellowLED, LOW);
  digitalWrite(RedLED, HIGH);// turn on led red when gate is closed
  delay(3000);

   MyServo.detach();
}

void OpenReservoir2()
{

  MyServo.attach(12);
  delay(100);
  Sleep_OK = false;
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print("feeder is opening");
  delay(1000);
  lcd.clear();

  digitalWrite(RedLED, LOW);
  digitalWrite(YellowLED, HIGH); // turn  on Yellow led while  gate is opening

  // sets the servo position according to the scaled value
  for (ServoPosition = 0; ServoPosition <= 75; ServoPosition += 1)
  { // goes from 0 degrees to 180 degrees steps of 1 degree

    MyServo.write(ServoPosition);              // tell servo to go to position in variable 'pos'
    delay(20);                       // waits 15ms for the servo to reach the position

    Feeder_Open = true;
```

```
  }

  digitalWrite(YellowLED, LOW);
  digitalWrite(GreenLED, HIGH); // turn on green led when gate opened

  lcd.setCursor(0, 0);
  lcd.print("Feeder is Opened");
  delay(3000);
  lcd.clear();
}


//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++3

void CloseReservoir3()
{
  Sleep_OK = false;
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print(" food dispensed ");
  // ServoPosition = 180; // set degree to 180
  digitalWrite(GreenLED, LOW);
  digitalWrite(YellowLED, HIGH); // turn  on Yellow led while  gate is closing
  for (ServoPosition == 75; ServoPosition >= 0; ServoPosition -= 1)
  { // goes from 180 degrees to 0 degrees
    MyServo.write(ServoPosition);              // tell servo to go to position in variable 'pos'
    delay(25);                      // waits 15ms for the servo to reach the position

    Feeder_Close = true;
    previousTime =  currentTime;
    previousTime2 =  currentTime2;
    previousTime3 =  currentTime3;

  // reset timers each day
     if ( currentTime == 1440)
     {
      currentTime = 0;
     }
```

```
    if ( currentTime2 == 1440)
    {
     currentTime2 = 0;
    }
    if ( currentTime3 == 1440)
    {
     currentTime3 = 0;
    }
  }
  digitalWrite(YellowLED, LOW);
  digitalWrite(RedLED, HIGH);// turn on led red when gate is closed
  delay(3000);

   MyServo.detach();
}

void OpenReservoir3()
{

  MyServo.attach(12);
  delay(100);
  Sleep_OK = false;
  lcd.clear();
  lcd.setCursor(0, 1);
  lcd.print("feeder is opening");
  delay(1000);
  lcd.clear();

  digitalWrite(RedLED, LOW);
  digitalWrite(YellowLED, HIGH); // turn  on Yellow led while  gate is opening

  // sets the servo position according to the scaled value
  for (ServoPosition = 0; ServoPosition <= 75; ServoPosition += 1)
  { // goes from 0 degrees to 180 degrees steps of 1 degree

    MyServo.write(ServoPosition);              // tell servo to go to position in variable 'pos'
    delay(25);                       // waits 15ms for the servo to reach the position

    Feeder_Open = true;
```

```
  }

  digitalWrite(YellowLED, LOW);
  digitalWrite(GreenLED, HIGH); // turn on green led when gate opened

  lcd.setCursor(0, 0);
  lcd.print("Feeder is Opened");
  delay(3000);
  lcd.clear();14qe
}

void Get_Distance()
{
  // grab distance from the sensor and convert it to cm

  RAWdistance = sonar.ping_median();
  distance = sonar.convert_cm(RAWdistance);

  lcd.setCursor(0, 1);
  lcd.print(distance);
  lcd.setCursor(3, 1);
  lcd.print("Cm");
  delay (200);
}

void Go_TO_Sleep()
{
  Sleep_OK = true;
  Wake_UP_OK = false;
  noInterrupts (); // Disable interrupts before entering sleep
  //attachInterrupt (INT2, wake, FALLING);  // Wake on falling edge of D3
  interrupts (); // Enable interrupts to ensure next instruction is executed
  lcd.clear();
   lcd.setCursor(0, 0);
  lcd.print("sleep");
  delay(1500);
  lcd.noBacklight(); // turn on backlight.
  Serial.flush();
   set_sleep_mode(SLEEP_MODE_PWR_DOWN); // setting the sleep mode / full sleep

  sleep_cpu(); // activate sleep mode
```

```
  digitalWrite(13, LOW);
  digitalWrite(GreenLED, LOW);
  digitalWrite(YellowLED, LOW);
  digitalWrite(RedLED, LOW);

  delay(30000);

  }


  void WakeUp()
  {
    Wake_UP_OK = true;
    Sleep_OK = false;
     detachInterrupt (INT2); // Disable interrupts on D10
     sleep_disable(); // Disable sleep mode
      Serial.flush();


  }

  void setRTCagain()
{

 if ( RTC.alarm(ALARM_1) )
  {
    //Serial.println("Now Alarm!");
    // set the alarm

     lcd.clear();


    DateTime now = rtc.now();
    DateTime future (now + TimeSpan(0, 0, 0 ,30)); //Days, Hours, Minutes, Seconds
   // Serial.println(F("Alarm is set to: "));
    printDateTime(future);
    RTC.setAlarm(ALM1_MATCH_DATE, future.second(), future.minute(), future.hour(),
future.day());

    digitalWrite(13, HIGH);
    digitalWrite(RedLED, HIGH);
```

```
   lcd.backlight(); // turn on backlight.
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Awake...");
  delay(1000);
  Sleep_OK = false;
  Wake_UP_OK = true;
 }
}

// display alarm time
void printDateTime(DateTime t)
{
 lcd.setCursor(14, 0);
  lcd.print((t.hour() < 10) ? "0" : "");  lcd.print(t.hour(), DEC);   lcd.print(':');
    lcd.print((t.minute() < 10) ? "0" : "");   lcd.print(t.minute(), DEC);   lcd.print(':');
    lcd.print((t.second() < 10) ? "0" : "");   lcd.print(t.second(), DEC);   lcd.print('/');
     delay(1000);
}

// display day time
 void updateDisplay()
  {
  tmElements_t tm;
  RTC.read(tm);

  // Print date and time

  lcd.setCursor(12, 2);
  lcd.print(tm.Month);
  lcd.print("/");
  lcd.print(tm.Day);
  lcd.print("/");
  lcd.print(tmYearToCalendar(tm.Year)-2000);
  lcd.setCursor(12, 3);
  lcd.print(tm.Hour);
  lcd.print(":");
  lcd.print(tm.Minute);
  lcd.print(":");
  lcd.print(tm.Second);
  delay(1000);
```

Pet Feeder with Pet Identification

```
}

void Time_Interval_1()
{


        currentTime = millis();
        currentTime = (currentTime / 60000);
        previousTime = (previousTime / 60000);


}

void Time_Interval_2()
{
        currentTime2 = millis();
        currentTime2 = (currentTime2 / 60000);
        previousTime2 = previousTime2 / 60000;
}


void Time_Interval_3()
{
        currentTime3 = millis();
        currentTime3 = (currentTime3 / 60000);
        previousTime3 = (previousTime3 / 60000);
}
```